Time-Dependent Recommender Systems for the Prediction of Appropriate Learning Objects

vorgelegt von

M.Sc.

Christopher Krauß

geb. in Berlin

von der Fakultät IV – Elektrotechnik und Informatik
der Technische Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
- Dr.-Ing. -

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender:	Prof. Dr. Axel Küpper
Gutachter:	Prof. Dr. Manfred Hauswirth
Gutachterin:	Prof. Dr. Agathe Merceron
Gutachter:	Prof. Dr. Hendrik Drachsler

Tag der wissenschaftlichen Aussprache: 29. Mai 2018

Berlin 2018

Abstract

This dissertation deals with adaptive learning technologies which aim to optimize *Technology Enhanced Learning (TEL)* offerings to fit the individual learner's needs. Thereby, *Recommender Systems* play a key role in supporting the user's decision process for items of interest. This works very well for e-commerce and *Video on Demand* services. However, it is found to be the case that these traditional *Recommender Systems* cannot be directly transferred to *TEL* as the recommendation of course items follows a particular educational paradigm. The special conditions of this paradigm are first investigated and then taken into account for the realization of new algorithms.

In order to allow a broad interoperability of a *Recommender System* with other technical components, a set of open standards and specifications results in a reference architecture for such an adaptive learning environment. Based on the realized architecture, activity data have been collected from students using course materials available online – the courses themselves comprising face-to-face lectures backed up by digital representations of the presented contents, blended learning settings as well as online-only courses. The courses provided access to the course materials via a novel *Learning Companion Application*. This app also presents learning recommendations to make the content selection more efficient and effective.

Thereby, this work indicates that an educational *Recommender System* should not be evaluated using standard evaluation frameworks that utilize, for instance, a classical n-fold cross-validation. For this reason, a time-dependent evaluation framework is defined to investigate the precision of the Top-N learning recommendations at various points in time. Moreover, a new measure is introduced to determine the *Mean Absolute Timeliness Deviation* between an item recommendation and the time when it is actually accessed by the user.

Subsequently, four major techniques for *Recommender Systems* are realized and applied to the collected data, evaluated with the time-dependent evaluation framework and successively optimized. As a reference implementation, a traditional *Collaborative Filtering* algorithm is developed and extended to incorporate time information. The results are then compared to the results of other time sensitive algorithms: an *Item-based Collaborative Filtering* approach which has previously been applied to *TEL* and a new learning path generator which incorporates a set of contextual information. Finally, a novel time-weighted *Knowledge-based Filtering* algorithm is presented and exhaustively analyzed. The evaluation results indicate that time-dependent filtering which incorporates multi-contextual activity data can produce the most precise recommendations.

Zusammenfassung

Die vorliegende Dissertation beschäftigt sich mit adaptiven Lerntechnologien, die sich an die individuellen Bedürfnisse der Lernenden anpassen. Dabei spielen vor allem Empfehlungssysteme eine Schlüsselrolle, da sie den Entscheidungsprozess der Benutzer unterstützen. Das funktioniert sehr gut für *E-Commerce* und *Video on Demand*-Dienste. Allerdings können diese Mechanismen nicht einfach für den Bereich des Technologie-gestützten Lernens übertragen werden, da die Empfehlungen von Kursinhalten einem sehr speziellen Paradigma folgen. Die Eigenschaften dieses Paradigmas werden in der Dissertation erst analysiert und anschließend als Basis für neue Algorithmen berücksichtigt.

Um eine breite Interoperabilität des Empfehlungssystems mit anderen technischen Komponenten zu gewährleisten, wurden offene Standards und Spezifikationen umgesetzt, mit deren Hilfe eine Referenzarchitektur für adaptive Lernumgebungen umgesetzt wurde. Basierend darauf wurden Aktivitätsdaten in Echtwelt-Kursen gesammelt – von Präsenzunterricht, welcher durch digitales Vorlesungsmaterial unterstützt wurde, über Blended Learning-Umgebungen bis hin zu ausschließlichen Online-Kursen. Alle Kursteilnehmer hatten Zugriff auf die Kursmaterialien über die Lernbegleiter-App. Der Entscheidungsprozess der Lernenden wurde durch ein Lernempfehlungssystem unterstützt.

Dabei hat sich herausgestellt, dass herkömmliche Evaluationstechniken, wie die *n-Fold Cross-Validation*, nicht für die Evaluation von Lernempfehlungssystemen geeignet sind. Deshalb wurde ein zeitabhängiges Evaluations-Framework definiert, mit dem die Präzision von Top-N-Lernempfehlungen zu verschiedenen Zeitpunkten analysiert werden kann. Zusätzlich wurde eine neuartige Messgröße eingeführt, die „*Mean Absolute Timeliness Deviation*", die den zeitlichen Abstand zwischen Empfehlungen und dem späteren Abruf der Inhalte durch den Benutzer misst.

Darauf basierend konnten vier Haupttechniken für Empfehlungssysteme realisiert und auf die gewonnenen Datensätze angewandt werden. Dann wurden diese mit dem definierten Evaluations-Framework ausgewertet und sukzessive optimiert. Als Referenzimplementierung diente ein traditioneller *Collaborative Filtering*-Algorithmus. Dieser ließ sich mit zeitabhängigen Algorithmen vergleichen: mit einer *Item-based Collaborative Filtering*-Methode, welche bereits für das Technologie-gestützte Lernen angewandt wurde, sowie mit einem Lernpfad-Generator, der kontextabhängige Informationen verarbeitet. Anschließend ist ein neuartiger kontextsensitiver und zeitabhängiger *Knowledge-based Filtering*-Algorithmus vorgestellt und ausgewertet worden. Die Arbeit zeigt, dass die präzisesten Empfehlungen durch zeitabhängige Filter-Algorithmen produziert werden, die zusätzlich mehrere Typen von Aktivitätsdaten verarbeiten.

Acknowledgments

Exam Committee & Supervisors

First of all, I would like to thank my doctoral thesis supervisor, Prof. Dr. Manfred Hauswirth (chair of *Open Distributed Systems* at the *Technische Universität Berlin*), for his support and guidance as well as for providing feedback regarding the completion of the dissertation. He is not only my supervisor, since as the Executive Director of the *Fraunhofer-Institute for Open Communication Systems (FOKUS)*, he is also the primary enabler of my research. Without him, this work would not have been possible.

I am especially indebted to Prof. Dr. Agathe Merceron of the Beuth University of Applied Sciences, who supported me in writing a doctoral thesis on educational *Recommender Systems*. She helped with the structuring of the dissertation and shared her detailed knowledge on *Data Mining* and *Technology Enhanced Learning*. She understood the issues raised during the course of my study, structured my ideas and made practical suggestions as this work was carried out.

I would like to offer my special thanks to Prof. Dr. Hendrick Drachsler of the Goethe University Frankfurt and the German Institute for International Educational Research, who agreed to examine this dissertation. As one of the most prominent researchers in the field of *Recommender Systems* for *Technology Enhanced Learning*, he laid substantial groundwork for this doctoral thesis.

Moreover, I would like to thank Prof. Dr. Axel Küpper of the research group *Service-centric Networking* at Telekom Innovation Laboratories (An-Institut *Technische Universität Berlin*) for his willingness to support my doctoral application and the commitment to chair the doctoral committee.

I am grateful for the encouragement given to me by Dr. Stefan Arbanowski who was not only my day-to-day supervisor but also my mentor. Regarding my research approach, he assisted me in thinking outside of the box and encouraged me to keep going. He offered me many opportunities to deepen my knowledge through engagement with the research community, attendance at scientific conferences and networking with industry.

Smart Learning Team

I also would like to thank the other colleagues who were involved in my various projects and who thus also enabled this work. First and foremost, I would like to thank the entire team of the Smart Learning project that produced some outstanding results. In particular, I would like to highlight the contributions of individuals from a number of organizations. From *FOKUS*: Miggi Zwicklbauer (who is also a patient officemate); from *Beuth University* under the leadership of Prof. Dr. Agathe Merceron: Sinh-Truong An and Francois Dubois; and from the *IZT*: Dr. Michael Scharp.

In addition, many student research assistants supported my learning-related projects as employees

of *FOKUS*. Among others and in chronological order from the time they started work: Igor Fritzsch, Rakesh Chandru, Berken Bayat, Dominique Jürgensen, Jessica Gomez and Tolga Karaoglu.

Students & Participants

A lot of university students and participants from other institutions utilized the *Learning Companion Application* – even though it was still in an early state. Their feedback was invaluable and the suggestions improved the project outcome significantly. I will not forget the large number of mainly anonymous test persons who invested their time and creativity in evaluating the app and taking part in the surveys.

Within the last 5 years, I have supervised eight Bachelor, Diploma and Masters Theses and more are in progress. Moreover, in excess of 50 students from the *Technische Universität Berlin (TU Berlin)* and Beuth University worked on the project topics I offered in order to earn module credits. These students demonstrated interesting directions for *Recommender Systems* and/or *Technology Enhanced Learning* – sometimes with excellent results.

Colleagues & Friends

I also would like to thank the many colleagues at *Future Applications and Media (FAME)* who I worked with in various other project contexts: Dr. Stephan Steglich, Robert Seeliger, Steffi Keller, Fabian Steinert, Stefan Pham, Louay Bassbouss, Sascha Braun, Daniel Silhavy and Görkem Güclü, to mention only a few.

I must also acknowledge my friends. Thank you for listening and for offering me advice. Special thanks go to Dr. Duong Nguyen-Thi and Carsten Schramm, Dr. Julia Rosenlöcher and Karsten Mesow, Susanne and Thorsten Helbig, Dr. Elisa Degenkolbe and Stefan Hartmann, Johannes Wollschläger and Paul Hecht as well as Berivan Kara and Marcel Melcher, Eric Stark and Felix Hahn – and, of course, to all their lovely children.

Family

I owe many thanks to Birgit and Jörg Karpow, Nicole and Lars Kunze as well as to their kids, Fenja and Felix. Moreover, I would like to thank the ”Hamburgers”, the ”Brandenburgers”, the ”Spanish”, the ”Engelhardts” as well as all the other members of my great family.

My gratitude is especially extended to Bryan Krauss, who is my brother and friend: thank you for the numberless hours of seriousness, distraction and fun.

It is an honor for me to thank my parents, Manuela and Torsten Krauss who have supported me in so many ways. They have influenced my career aspiration and professional development. Moreover, they always helped with words and deeds: I always knew that you believed in me and wanted the best for me.

Last, but certainly not least, it gives me great pleasure to thank my wife, Angela Krauss – who supported me in writing this dissertation in particular and who also encouraged me in every respect. Thanks also go to my son, Jonathan Krauss, who can always animate me with his high spirits and cheer me up with his smile. Since, as employee of Fraunhofer writing a dissertation is typically seen as free time activity, both suffered the long working hours of a husband and father whose mind was too often on the dissertation instead of the family. Without your backing, this journey would not have been possible: I am forever grateful. Thank You!

Preface

Motivation

During the realization of this dissertation, I talked to a lot of people – learners, teachers, content creators and service providers – and asked them for their opinion of our adaptive learning solutions and whether they found them acceptable. While I received a lot of motivating feedback from the educational staff, I had the most compelling conversions with the students on the *Advanced Web Technologies (AWT)* course at the *Technische Universität Berlin (TU Berlin)*. At the *TU Berlin* as well as at the Beuth University I regularly present some of my work on specialized topics in guest lectures, including on *Recommender Systems* and Data Mining. At the beginning of the lecture term, students frequently asked whether our course offering would fit their ideas on deepening their skills in the area of web development and personalization. Theses students often have some basic knowledge of *Hypertext Markup Language (HTML)*, *Cascading Style Sheets (CSS)* and *JavaScript (JS)*, but cannot imagine the technological scope of this course. By way of an answer, I would briefly introduce our *Learning Companion Application* where the taught concepts can be directly experienced through a real-world example. With the *Learning Companion Application (LCA)*, the course participants learn the theoretical background to advanced web programming and at the same time, observe how those can be realized in a web application. These explanations go down well with the students and result in a lot of positive feedback. Particularly interesting appears to be the idea of an application that adapts to the learners' needs, visualizes their estimated knowledge levels and recommends appropriate learning materials through our *Smart Learning Recommender (SLR)*. The course enrollment significantly increased since the *Learning Companion Application* was introduced, and at the same time the drop-out rate was reduced for this course. Even the average mark for the final test at the end of the course slightly improved[1].

While it is not possible for me to to evidence whether this outcome is the result solely of our adaptive platform (or is additionally affected by other circumstances), it shows the success of one of the primary goals of this work: learners become motivated because they experience new ways of learning – via a platform that reacts to the learners' needs. It assists course participants in their *Self-Regulated Learning (SRL)* skills, helps in the understanding of individual studying behaviors and automatically responds to identified learning weaknesses. Thereby, the *Smart*

[1] A first course run without the *LCA* in the winter semester of the 2015/2016 academic year comprised 72 course enrollments of which 39 participated in the final test and reached a mark of 2.00 on average. In a second course run in the winter semester of 2016/2017, the *Learning Companion Application* was introduced for the first time. Thereby, 99 students (of 126 enrolled students) used the app of which 83 students participated in the final test and reached an average mark of 1.90. Finally, a third course run also utilized the *LCA* and while the course had not ended before the submission of this dissertation, 118 students were enrolled for the course in the winter semester 2017/2018 and 78 of these had already used the app by the end of the enrollment period.

Learning Recommender aims at rendering knowledge acquisition more efficient and effective, at the same time offering an exciting service with innovative interactive digital media.

Dissertation History

Since 2010, I have worked and researched in the area of *Recommender Systems*. At this point, I was employed as a student research assistant and developed a working *Collaborative Filtering* system that is able to identify on-demand videos on a university multimedia platform that fit well with particular user preferences. In 2012, I received my Master's degree (graded very good with distinction) with completion of my thesis titled "Personalized Recommendations to be displayed on SmartTVs" that was supervised by both colleagues of the Beuth University of Berlin and of the *Fraunhofer-Institute for Open Communication Systems (FOKUS)*. The latter hired me as a Research Engineer within the business unit *Future Applications and Media (FAME)*. Since 2013, I have worked there as a Senior Project Manager, undertaking a number of national and international projects. In 2014, two exciting opportunities came up which enabled me to deepen my knowledge of *Recommender Systems* and at the same time transfer this knowledge to a new application area. The first one was to work on a nationally funded project that has the goal of introducing digital media to vocational training via an adaptive user interface – the Smart Learning project. The second project was part of the Software Campus program which aims at supporting high potential Ph.D. candidates in the realization of self-selected research topics. Both projects (and later a number of additional funded activities) enabled me to undertake research on a very focused problem, the problem which has led to this dissertation[2]. Moreover, they helped me to network with experts within the academic and industrial communities and to publish my findings[3].

[2] An overview of the projects and other activities which enabled this dissertation is presented in Appendix B.1.
[3] A list of relevant publications and other related activities of the author are presented in Appendix A.

Contents

List of Figures

List of Tables

List of Hypotheses

List of Abbreviations

ADL Advanced Distributed Learning Consortium
AICC Aviation Industry Computer-based Training Committee
AJAX Asynchronous JavaScript and XML
ANN Artificial Neural Network
API Application Programming Interface
AVMSD Audiovisual Media Services Directive
AWT Advanced Web Technologies
BDSG German Bundesdatenschutzgesetz
CAM SCORM Content Aggregation Model
CARS Context-Aware Recommender System
CbF Content-based Filtering
CBR Case-based Reasoning
CC Common Cartridge Specification
CDN Content Delivery Network
CF Collaborative Filtering
CLM Common Learning Middleware
CMS Content Management System
CP Content Packaging Specification
CSS Cascading Style Sheets
DAO Data Access Object
DF Demographic Filtering
FAME Future Applications and Media
FOKUS Fraunhofer-Institute for Open Communication Systems
GDPR General Data Protection Regulation
GUI Graphical User Interface
HACP HTTP AICC Communication Protocol
HF Hybrid Filtering
HTML Hypertext Markup Language
IbCF Item-based Collaborative Filtering
ILT Intelligent Learning Technologies
IMS Instructional Management System Global Learning Consortium
ISIS Information System for Instructors and Students

ITS Intelligent Tutoring Systems
IZT Institut für Zukunftsstudien und Technologiebewertung
JS JavaScript
JSON JavaScript Object Notation
KbF Knowledge-based Filtering
KNN K-Nearest-Neighbour
KPI Key Performance Indicator
LCA Learning Companion Application
LCMS Learning Content Management System
LLL Life-Long Learning
LMS Learning Management System
LO Learning Object
LOM Learning Resource Meta-data Specification
LRMI Learning Resource Metadata Initiative
LRS Learning Record Store
LTI Learning Tools Interoperability Specification
LU Learning Unit
MAE Mean Absolute Error
MATD Mean Absolute Timeliness Deviation
ML Machine Learning
MOOC Massive Open Online Course
OER Open Educational Resources
ODS Open Distributed Systems
OLX Open Learning XML
PDF Portable Document Format
PF Preference Filtering
PLE Personal Learning Environment
QTI Question and Test Interoperability Specification
RDF Resource Description Framework
RE Recommender Engine
REST Representational State Transfer
RMSE Root Mean Square Error
RMSTD Root Mean Square Timeliness Deviation
RS Recommender System
RTE SCORM Run-Time Environment
SCORM Sharable Content Object Reference Model
SLR Smart Learning Recommender
SN SCORM Sequencing and Navigation
SNA Social Network Analysis

SRL Self-Regulated Learning
SVM Support Vector Machine
TARS Time-Aware Recommender System
TBRA Time-based Recommender Approach for Lecture Materials
TEL Technology Enhanced Learning
TMG German Telemediengesetz
TU Berlin Technische Universität Berlin
UbCF User-based Collaborative Filtering
UbF Utility-based Filtering
URI Uniform Resource Identifier
URL Uniform Resource Locator
VoD Video on Demand
W3C World Wide Web Consortium
WWW World Wide Web
xAPI Experience API
XML Extensible Markup Language

1. Introduction

"We envision that by 2030, environments will provide truly individualized learning (optimized for the individual) capable of being completely adaptive and adaptable to a sufficiently complete representation of the learner (user model) in order to deliver the most optimized learning experience."
Woolf et al. (2010) [284, p. 60]

Various educational stakeholders have identified adaptive learning as one of the most promising and at the same time most challenging trends for *Technology Enhanced Learning (TEL)* (cf. [284, 270, 106, 195, 103]). An adaptive system is a digital environment that changes its presentation interface, its navigation flow and/or its content offering according to its perception of the user's needs [49]. This is a well-researched area and is established within the media web sector – including the e-commerce domain or streaming services for music and movies [209]. *Recommender Systems* are designed for users who lose track of the vast quantity of media and products available and thus need assistance with their content selection [234, p. 158]. In contrast to traditional search engines, a *Recommender System (RS)* does not need explicit input and search terms from its users [146], instead it supports users even when they themselves do not know what they are actually looking for. While this approach also sounds reasonable for the domain of *Technology Enhanced Learning*, the research in this area is still in the very early stages (cf. [191, 94, 92, 269]).

1.1. The Need for Adaptability in Educational Environments

Technology Enhanced Learning platforms utilizes digital learning materials for a variety of educational approaches and settings. In addition to the traditional ways of formal learning, *TEL* enables new paradigms, which allocate a more critical role to *Self-Regulated Learning* – especially in entirely or partly informal learning settings [94]. Instructors are less involved in these informal settings and thus more responsibility is required from the learners regarding self-motivation, self-control and time management. That is why a learning platform should adapt to the learners needs[4]. Verbert et al. note that in *TEL* "each learner uses her own tools, methods, paths, collaborations, and processes. Consequently, guidance within the learning process must be personalized to an

[4] In 2017, educational experts rated adaptive learning as the second most important trend for digital education [195]. Appendix B.3 introduces the results of this study.

extreme extent" [270, p. 319]. An educational *Recommender System* is crucial when the amount of content choices might overwhelm the user. "The fact that some choice is good doesn't necessarily mean that more choice is better. [... There] is a cost to having an overload of choice" [241].

"It is expected that personalised learning has the potential to reduce delivery costs, to create more effective learning environments and experiences, to accelerate time to competence development, and to increase collaboration between learners. Recommender Systems are one of the promising technologies to support people in finding the most suitable information and peer learners."
Drachsler et al. (2010) [92, p. 2850]

Typical Tasks

A lot of research has been done on *Recommender Systems* for supporting learners (e.g., [21, 231, 270]). However, similar computing tasks may also assist teachers as adaptive learning technologies can allocate the tasks of traditional educational staff to software components [106][5]. According to Ifenthaler and Schumacher, the use of adaptive learning platforms generates data which lead to conclusions on four different meta-levels [135, p. 177]:

1. The individual learner's level for adapting the environment to the personal learning process.
2. The course level for adjustments and content optimization by educators and content creators.
3. The institutional level relating to, for example, decisions across all offered courses.
4. The political level for inferences on the whole educational system.

In general, adaptive learning platforms enable the collection and processing of the learner's activity data which should lead to focused conclusions. The effect and accuracy of such algorithmic findings for learning environments differ in each topical area, course setting and even for each learner [269, 103]. Thus, these components can be seen as support tools to make the processes learning and teaching more personalized and comfortable as well as more effective and more efficient [135, p. 181].

1.2. Problem Statement

"We believe that Collaborative Filtering and rules can do for Learning Objects what Google did for the Web. The challenge will be to meet the real (collaborative and inferential) needs expressed by course creators and students."
Lemire et al. (2005) [174, p. 187][6]

This dissertation focuses on recommendations given in a closed-corpus setting, which means that the learning environment adapts to the learners' needs within a self-contained course [191, p.

[5] An analysis of the most important educational tasks (published by Darabi et al. [78]), and how adaptive learning technologies can assist them, can be found in Appendix B.4.

[6] Lemire is the inventor of the Slope One algorithm which will be extended to fit the needs of *TEL* and analyzed in this dissertation.

11 - 12]. The user interface offers content recommendations at particular points in time that fit with the current needs and the contextual situation of individual course participants. The most important task of the *Recommender System* is to support learners in achieving their personal learning goals. Personal goals are mainly related to formal assessments, but can additionally refer to informal objectives, such as the acquisition of a particular skill or piece of knowledge. Thus, an "appropriate" recommendation aims at making learning more efficient and effective by supporting the content selection process [220, 264]. This work shows that the generation of recommendations within an educational environment is a highly time-dependent problem that requires various contextual data.

Hypotheses

The first half of this dissertation introduces different concepts and related work and defines as a consequence a list of hypotheses. The overall scientific question which relates to the title of the dissertation is presented as follows:

> Scientific Question /SQ/:
> **How can a *Recommender System* predict appropriate items in a closed-corpus course environment?**

Thereby, the important terms, especially those that need further explanations (such as "appropriate") are defined in following chapters. For clarity, the formulation of the scientific question (/SQ/) and scientific hypotheses (/SH#/) follow Balzert's requirements syntax [27, p. 493-496]. A scientific hypothesis represents an educated guess regarding a causal relationship [265, p. 179-185] that is of interest for solving the corresponding scientific question and these will be discussed in the final chapters of this work.

1.3. Scope of this Dissertation

Key Contributions

This work combines two different research areas within the *Recommender System* domain: *Time-Aware Recommender Systems* and *Technology Enhanced Learning*. It demonstrates that well-known *Recommender System* approaches cannot be directly transferred to this kind of specialized application area. Established *Collaborative Filtering (CF)* and *Content-based Filtering (CbF)* algorithms, as well as commonly applied evaluation procedures, are not appropriate for the educational domain [94]. Moreover, the dissertation uncovers a lack of academic open datasets which leads to the necessity of collecting important activity data from real-world courses. Finally, a *Recommender System* represents only a part of a complex architecture consisting of different education-oriented components. The *RS* must communicate with other technical components via well-defined interfaces and exchange common data formats. The key contributions of this dissertation are formulated as follows:

Contribution 1 is the definition of a **specialized paradigm for educational *Recommender Systems***

that aim at making learning more effective and more efficient. While traditional systems rely on preferences, an adaptive learning system should support learners in their *Self-Regulated Learning* skills and thus it must depend on the user's learning need.

Contribution 2 comprises the design, realization and evaluation of existing and **novel *Time-Aware Recommender Systems***. These algorithms are applied for closed-corpus courses in *TEL*. Thereby, diverse methods have been realized, adjusted and iteratively optimized in various settings. The core approaches comprise the Slope One algorithm [175], a time-aware educational *Recommender System* [126], a new time-dependent Learning Path Generator and a self-devised *Knowledge-based Filtering* approach, the *Smart Learning Recommender* [161].

Contribution 3 includes the design and application of a **reliable evaluation framework** for *Time-Aware Recommender Systems* in general, and *RSs* in *Technology Enhanced Learning* in particular. Moreover, a self-devised measurement value – the *Mean Absolute Timeliness Deviation (MATD)* – is introduced to determine the timely nature and accuracy of the recommendations.

Contribution 4 consists of the concept and realization of a **technical reference architecture** for different educational settings. The architecture allows data processing techniques to be used to offer learning-oriented adaptive services and time-dependent personalization features in a re-usable way for various stakeholders. This comprises the definition and selection of **appropriate interfaces and metadata formats**.

Contribution 5 is the organization of **real-world courses** utilizing the realized technical infrastructure and the **collection of learning activity data** for the evaluation of the developed algorithms. The standardized metadata formats are field tested for utilization through time-dependent educational *Recommender Systems*.

To achieve these goals, an exhaustive analysis of existing concepts, realized approaches and applied algorithms is conducted that leads to the identification of the gaps in previous work and the requirements for current and future research. At the end of this dissertation, the different algorithmic approaches are compared and the critical outcomes of the evaluations are discussed.

1.4. Organization of the Dissertation

An introduction to the citation and formatting styles that should assist in providing clarity to the text can be found in Appendix B.2. The remainder of the dissertation is structured as presented in Figure 1.1 and below.

Chapter two starts with a presentation of the necessity of employing *Recommender Systems* in adaptive learning environments and explains the state of the art of typical filtering algorithms. Thereby, approaches that build the bases for further developments are introduced and classified. Moreover, typical issues and challenges that also have an impact on the area of *Technology*

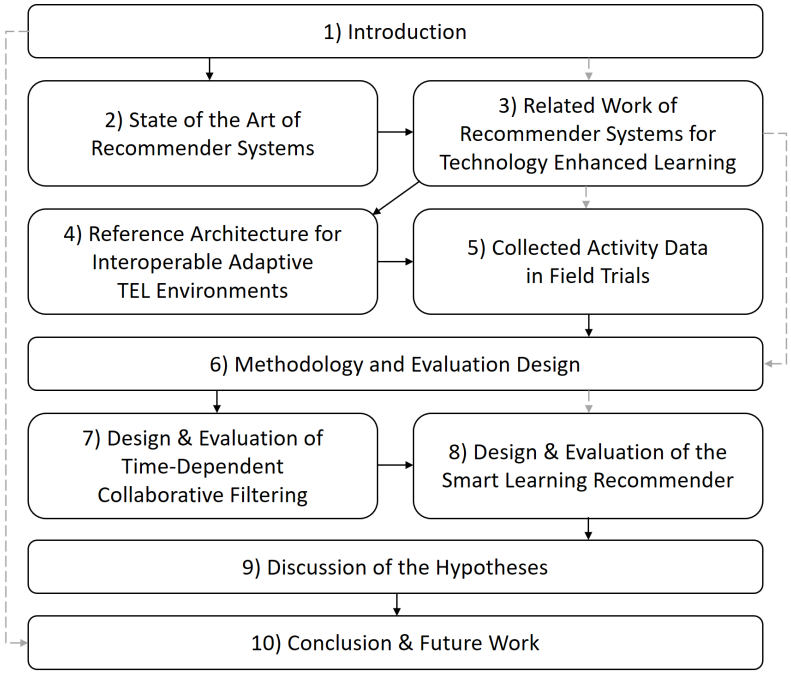

Figure 1.1.: Organization of the dissertation, where the arrows represent the recommended reading paths: black arrows show the document structure and gray arrows indicate possible shortcuts.

Enhanced Learning are discussed. Chapter three defines a special paradigm which must be followed by educational *Recommender Systems* that recommend course items. The paradigm's character and its aspects are discussed through the analysis of related works in the field of *TEL Recommender Systems*, *Context-Aware Recommender Systems (CARSs)* and *Time-Aware Recommender Systems*. Chapter four defines the current standards as well as the established specifications which build the basis for the realized reference architecture of an adaptive learning environment. Based on this architecture, different real-world courses are conducted and the learners' activity-related data are collected to form three major datasets which are described in Chapter five. The methodological approach of this work is presented in Chapter six and comprises a time-dependent evaluation framework and the introduction of common measures as well as the new measure, the *Mean Absolute Timeliness Deviation*. Chapter seven contains the evaluation and comparison of different *Collaborative Filtering* and *Time-Aware Recommender System* algorithms that are applied to the collected data. With the *Smart Learning Recommender*, a distinct algorithm is evaluated in Chapter eight that follows in all respects the educational paradigm previously defined. Thereby, the *SLR* focuses on the incorporation of context and time information. The hypotheses are discussed in Chapter nine taking the outcomes of the evaluations into account. Finally, this work concludes with a summary and an outlook.

2. State of the Art of Recommender Systems

Adaptive Systems

A web system is defined as adaptive "if it is able to change its own characteristics automatically according to the user's needs" [205]. This feature is called personalization ability as it adapts to the user's individual conditions. According to Brusilovsky et al. [49], adaptive systems can be classified according to the three main adaption technologies: "Adaptive Content Selection", "Adaptive Navigation Support" and "Adaptive Presentation". A *Recommender System (RS)*, which corresponds to a special shape of an adaptive system, may help, particularly for content selection, to adapt a web service to the user's preferences and needs. However, it may also recommend better (digital or analog) navigation strategies and predict the most appropriate interface to present to the end user.

About this Chapter

This chapter introduces basic concepts of *RS*s, different approaches that may be used to classify them and the main recommender techniques which form the basis for the design of the new methods discussed in this work. However, this chapter also focuses on typical metadata and discusses the main challenges and issues in *Recommender Systems*.

2.1. Definition of Recommender Systems

RS as Advisor

As the field of *Recommender Systems* has been researched for a number of decades[7], it has multiple definitions. Most of these describe the overall task to assist the user in identifying relevant items among a huge amount of available items, where the user is not able to analyze the appropriateness of each [234, p. 158]. In contrast to search engines, *Recommender Systems* do not necessarily need any explicit input data, such as search terms. Instead, similar to an educational staff member, it acts as an intelligent advisor, although in this case based on previously collected data instead of experience[8]. In a typical *RS* "people provide recommendations [related to their preferences] as inputs, which the system then aggregates and directs to appropriate recipients" [226]. Adomavicius and Tuzhilin [9] distinguish recommender systems from other related areas –

[7] A brief history of *Recommender Systems* is presented in Appendix B.5.

[8] The ten most common tasks of *Recommender Systems* in general as well as three particular *RS* tasks for the educational domain are listed in Appendix B.6.

such as cognitive science, approximation theory, information retrieval or forecasting theories – by the main recommender problems that "explicitly rely on the rating structure" – even though the users do not see any ratings. In contrast to the rating prediction task, another common definition focuses on the top-N recommendation task [57, p. 71] which better fits the task of an educational adaptive system. A *Recommender System* delivers a list of recommended items that may be relevant for the user with the help of a numeric score – the relevance score – instead of ratings [242, p. 2].

Important Terminology

The terms *Recommender System* and *Recommender Engine (RE)* are often used in the same context, but they in fact have slightly different meanings. These meanings are defined for this work as follows: A *RS* (also Recommendation System) is the overall system that works as an independent service for the end user. Technically, it consists of the whole needed infrastructure: user interfaces, service logic layers and databases as well as the mined data and algorithm models. *RSs* are "software tools and techniques providing suggestions for items to be of use to a user" [146] (also mentioned in [187], [53], [226]). The *Recommender Engine* (also Recommendation Engine or Engine), in turn, represents the set of technical components needed for the calculation of recommendations. In most cases, only the core components, such as the algorithms and datasets, are implied here.

Application Areas

Park et al. [209, p. 10063] analyzed the main application areas in the literature, as shown in Figure 2.1. Among the 46 journals studied, movie recommendations, as known from *Video on Demand (VoD)* services such as Netflix, represent the most researched domain, followed by shopping recommendations, relating to e-commerce services such as Amazon[9]. The category "others" represents niche item types (including for instance learning materials) as well as work on generic approaches that are transferrable to other application areas. As can be seen, *TEL Recommender Systems* play a minor role in the research literature compared to other item types.

Formal Definition

Most commonly, "the recommendation problem is reduced to the problem of estimating ratings for the items that have not been seen by a user" [9]. Therefore, the problem of recommendation prediction can be seen as a user- and item-dependent task[10], expressed by the following mathematical function [57, 9]:

$$F : U \times I \rightarrow R, \tag{2.1}$$

where U is a set of users and I is a set of items. F is an approximation function that aims to predict a set of valid ratings R. However, the set of R values does not need to be revealed to the end user and can represent feedback types other than ratings. For instance, it might be a Boolean value whether a user is interested in an item or not – or the predicted percentage a student requires to learn a specific item to pass the final exam. This dissertation uses the term

[9] The benefits of *Recommender Systems* from a business perspective are introduced in Appendix B.7.

[10] Appendix B.8 introduces the general user and item data as well as the different feedback types processed by *Recommender Engines*.

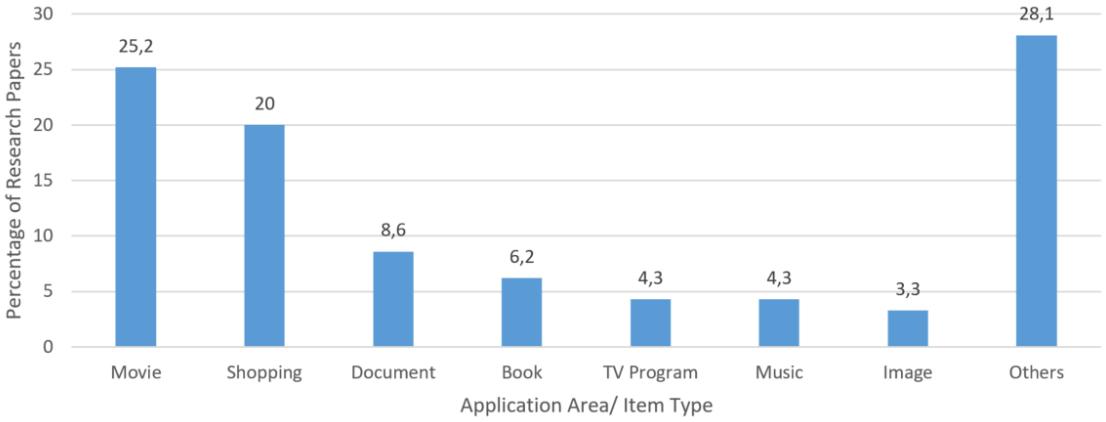

Figure 2.1.: Main Application Areas of Recommender Systems in Scientific Research (information taken from [209, p. 10063]).

relevance scores for the values of R, so that it can be used more generically. The rest of the equation is not affected.

Relevance Score

The validity of R depends on the allowed range of the *relevance scale* of the particular recommendation engine. For example, a five star rating might be in the range of [0,5] or [1,5] – depending on the meaning of the 0 value. Each relevance score $r_{u,i} \in R$ represents an element of feedback for user $u \in U$ on item $i \in I$. Predicted, non-real, scores are noted as $p_{u,i} \in PR$ [57].

Top-N List

In most cases, the user retrieves an ordered set of item predictions that is restricted to the n most relevant relevance scores, the so-called Top-N list. The items in that list can be represented by a list of relevance scores TN, where $TN \subseteq PR$ [86, 146]. Consequently, in a Top-10 list, the ten most relevant items are presented to the user ranked in order of their relevance.

2.2. Classification of Recommender Approaches

Filter Classes

There are multiple ways to classify recommender approaches. The two main concepts are *Content-based Filtering (CbF)*, where only the content of the items is compared, and *Collaborative Filtering (CF)*, which takes the feedback of other users into account [9, 53]. In addition, some other concepts exist which are classified in different ways in the literature. An overview of the different approaches is shown below:

- *Content-based Filtering (CbF)*: Recommendations are based on the features associated with items and the user ratings for these features [9, 53, 146].
- *Collaborative Filtering (CF)*: Recommendations are based on the rating histories of similar users [9, 53, 146]. *CF* can be categorized as *Item-based Collaborative Filtering (IbCF)* and *User-based Collaborative Filtering (UbCF)*.

- *Hybrid Filtering (HF)*: Recommendations are based on a combination of the other approaches [9, 53, 146].
- *Knowledge-based Filtering (KbF)*: Recommendations are based on the extracted understanding of the user's needs and preferences [53, 146].
- Additional Filtering Approaches: Other approaches, that cannot be directly classified as *CbF*, *CF* or *HF*, are out of scope for this work[11].

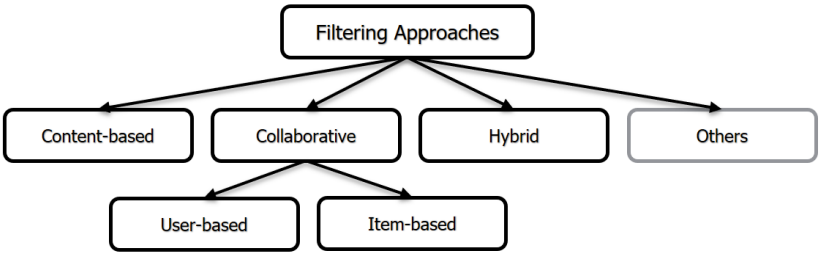

Figure 2.2.: Classification of main filtering approaches in *Recommender Systems*.

Hybrid *Recommender Systems*, in turn, aim at combining different filtering techniques to a single result set[12]. Figure 2.2 shows recommender classification taxonomy defined in this work.

2.2.1. Content-based Filtering

Content-based Filtering compares items based on their metadata. The most popular recommender techniques in this area are *Artificial Neural Network (ANN)*, Bayesian Classifiers, Clustering, Decision Trees and TF-IDF from information retrieval [9]. To create a mathematical understanding of the utilized metadata, the primary attribute types are introduced briefly. These attributes will also be the basis for all the other *RS* approaches in this dissertation.

2.2.1.1. Attribute Types

Similar to the classical variables in computer science, data mining techniques differentiate between a set of attribute types which make the attributes reusable and independent from a particular programming language. However, these attribute types can then be transferred and implemented as variables during the realization phase of an algorithm. Figure 2.3 shows the introduced attributes for recommender operations.

Ordered and Unordered Types

The general differentiation is between ordered and unordered types [122, p. 9-10]. Attributes of an ordered type can be allocated to a reasonable value and sorted in a meaningful manner.

[11] These rarely used approaches, such as Case-based Filtering [35], Community-based Filtering [176], Demographic Filtering [212], Learning-based Filtering [282], Preference Filtering [141], Rule-based Filtering [282], Utility-based Filtering [131] etc. will be handled as sub-classes of the most prominent filtering approaches. For the sake of completeness, these approaches are introduced in Appendix B.9.

[12] A representative example of *Hybrid Filtering* is the TV Predictor [167] which is introduced in Appendix B.10.

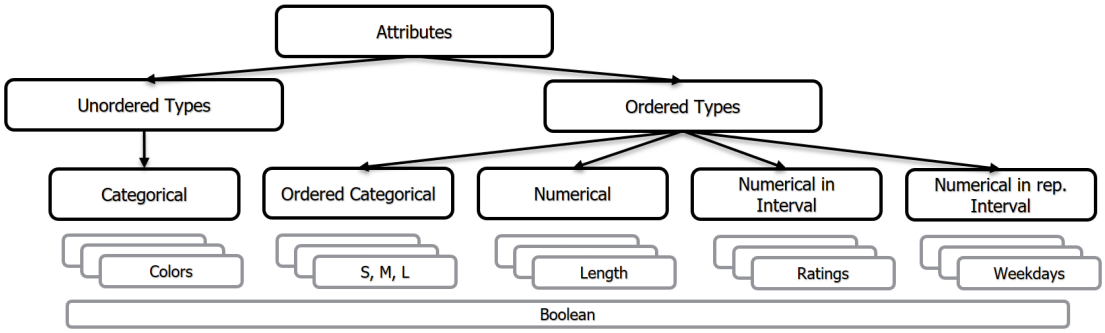

Figure 2.3.: Important attribute types in *Recommender Systems* (in black boxes) and examples (in gray boxes).

While this is the most common and important type of attribute in the area of data mining, some attributes cannot be sorted reasonably as they have only labels – so for instance for colors[13]. An ordered categorical type, in turn, can have a label, but also might be compared gradually with other elements of the same attribute[14]. Numerical values, in turn, might have an upper or lower bound, but this is not necessarily the case. A numerical is classified as a numerical without limitations or a numerical, which might be limited to a range, such as a one to five star rating system.

Repeating Interval

During the realization of different data mining projects, one additional type of attribute was missing – this attribute plays a crucial role for time-dependent systems. A time-dependent *Recommender System* needs to compare values of numerical time attributes in an interval (such as daytime, weekdays, years etc.) that repeats regularly. However, the distance between the upper and lower bounds should not be the highest possible distance, as for ratings for instance, but instead the lowest. Given the example of daytime, which is numerical within an interval. The first minute of the day (00:00 am) is very close to the last minute (11:59 pm or 23:59 on a 24 h range). In contrast, noon and midnight show the highest possible distance of 12 hours. The same applies to weekdays, months or seasons of a year. This attribute is called numerical in a repeating interval.

2.2.1.2. Similarity and Distance

Similarity Value

The term *Content-based Filtering* describes a filtering class that analyzes content and thus might simply compare metadata attributes of two elements (e.g., two items, two users or of a user and an item). The attributes can contain item or user descriptions, as well as item feedback of particular users. The similarity value is in the range $[0; 1]$: 1 represents an absolute equality

[13] Indeed, it is possible to allocate the frequency of the spectrum to color attributes, but in general, users only require to differentiate between them: ”red is not green and not yellow”. Categorical attributes are either the same or distinct.

[14] For instance, dress sizes, such as small, medium or large, can be related to a value and then be sorted.

between two attributes and 0 the highest possible diversity [252, p. 678-680].

$$sim(p,q) = 1 - \frac{|p-q|}{max(A) - min(A)}, \tag{2.2}$$

p and q are the numerical values taken by two elements of the same attribute. $max(A)$ is the highest and $min(A)$ is the lowest possible value of that attribute.

Cosine-based Similarity

For most applications, more than one attribute needs to be compared. Therefore, different metrics will be introduced, for example, the Cosine-based Similarity.

"The Cosine-based Similarity is used to calculate the similarity of two elements by treating them as vectors:

$$cSim(e_1,e_2) = cos(\vec{E_1},\vec{E_2}) = \frac{\vec{E_1}\cdot\vec{E_2}}{|\vec{E_1}|\times|\vec{E_2}|}, \tag{2.3}$$

e_1 and e_2 are the elements to be compared like items or users. $\vec{E_1}$ and $\vec{E_2}$ are vectors representing all [numerical] features of this element, so when n is the number of all attributes of an element, then $\vec{E_1} = (a_{1,e_1}, a_{2,e_1}, a_{3,e_1}, ..., a_{n,e_1})$. (cf. [64, p. 619], [245, p. 929])"

Cf. [167, p. 66][15].

Item-to-Item Mapping

Classical Item-to-Item Similarity Mapping (also called content-to-content mapping) represents an exception among filtering systems and is classified here as a particular case of *Content-based Filtering* – in fact, it is hard to classify at all because it can also be used to improve other filtering approaches [181]. Service providers might use *CbF* techniques to recommend similar items to the one that was just consumed. Using Cosine-based Similarity for instance, the two elements e_1 and e_2 are items that are compared by considering all their item features $\vec{E_1}$ and $\vec{E_2}$.

Distance Value

In contrast to the similarity value, the distance value is used when there is no limit of diversity. For the distance value, 0 means absolute equality but there is no upper bound. In general, the distance value can be calculated from the similarity value and vice versa (as long as the similarity value is restricted to [0;1]):

$$dist(p,q) = 1 - sim(p,q). \tag{2.4}$$

Euclidean Distance

An example comparison method using the distance value is the Euclidean distance $eDist$ that is the square root of the sum of the single numerical attribute distances (in this case the range is not restricted to [0;1]):

$$eDist(e_1,e_2) = \sqrt{\sum_{i=0}^{N} dist(a_{i,e_1}, a_{i,e_2})}. \tag{2.5}$$

Each attribute value a_{i,e_1} at position i of element e_1 is compared using the distance value to the according attribute value a_{i,e_2} of element e_2 [184, p. 305].

[15] This section is based on an algorithm used in the TV Predictor was written by Christopher Krauss and explains a well-established *Content-based Filtering* approach.

It is recommended to decide, for a single metric, the distance or the similarity, and to use this metric for the entire application area to avoid transferring errors. The approaches introduced in this dissertation will mainly focus on the similarity value as most attributes are in the fixed range [0;1].

2.2.1.3. Challenges for Content-based Filtering

Limited Content Analysis

Content-based methods require well-structured machine-readable data. Unstructured datasets might require pre-processing steps, therefore "in order to have a sufficient set of features, the content must either be in a form that can be parsed automatically by a computer (e.g., text) or the features should be assigned to items manually" [9, p. 737]. The same type of item information, e.g., the color of the product, must be tagged with the same keyword. While this sounds obvious, two item sets of different sources often show different feature structures and tags. Another challenge relating to content analysis emerges "if two different items are represented by the same set of features, they are indistinguishable" [9, p. 737].

Over-specialization and Portfolio Effect

When the *Recommender System* only recommends items "that score highly against a user's profile, the user is limited to being recommended items that are similar to those already rated" [9, p. 737]. Transferred to the problem of educational recommendations, learners might skip the study of particular topics and will then only receive a limited category of recommendations. The more the users obtain recommendations in one item category, the more feedback data are generated for this particular category. This intensifies the overspecialization. In some cases, a previously consumed item is recommended again – which represents a problem for items such as movies that are less interesting in terms of second consumption.

One solution is serendipity [134], where "items should not be recommended if they are too similar to something the user has already seen, such as different news article describing the same event". At the worst, the whole community might obtain only certain specific item recommendations, while the rest of the items will never be recommended. Some services compensate for these effects by suggesting a low percentage of random items in order to obtain even feedback on less rated products [52], [134].

New User Problem

Another big challenge occurs when a new user registers at a *CbF* system. "The user has to rate a sufficient number of items before a content-based recommender system can really understand the user's preferences and present the user with reliable recommendations. Therefore, a new user, having very few ratings, would not be able to get accurate recommendations" [9, p. 737]. Most services offer a set of items that need to be rated by the user before the service begins to offer recommendations. The selection process of appropriate first items is a particular *RS* challenge [13, 210, 146].

2.2.2. Collaborative Filtering

Collaborative Filtering means that "people collaborate to help one another perform filtering by recording their reactions to documents they read" [113] or any other item feedback. According to Adomavicius and Tuzhilin [9] commonly deployed *Collaborative Filtering* techniques are *ANN*, Bayesian Networks, Clustering, Graph Theory, Linear Regression and Probabilistic Models. All of these require a representation of the user feedback on consumed items in a so-called user–item matrix[16].

As *CF* represents the most common filtering technique, a lot of different approaches have been established to predict missing feedback values. These are introduced in the following.

> "To find the best fitting items for a user, there are some concrete approaches, such as Item-based Top-N Recommendation Filtering, that filters the items by including the ratings of all other users, and User-based Top-N Recommendation Filtering, that first searches for most similar users and retrieves their best rated items [180]. In contrast to Item-based Filtering, User-based Top-N Recommendation Filtering focuses on finding similar users – called neighbours. So the objective of Neighbourhood-based *Collaborative Filtering* is to find the nearest neighbour. Afterwards the Top-N items of the nearest neighbours are predicted"
>
> Cf. [163, p. 366][17].

The proceeding subsections introduce the two *Collaborative Filtering* approaches in detail: *Item-based Collaborative Filtering (IbCF)* and *User-based Collaborative Filtering (UbCF)*. The presented algorithms are utilized in this work.

2.2.2.1. Item-based Filtering

People tend to prefer similar items to the ones they have previously liked. The power of *Collaborative Filtering* comes from its ability to take every item rating of every user within the community into account. However, there is at some point a bottleneck: the higher the number of "neighbors among a large user population of potential neighbors" [235, p. 285], the more time consuming is the search for these neighbors. *Item-based Collaborative Filtering*, or Item-to-Item Collaborative Filtering, aims at identifying appropriate items based on the consumption information of other items. Thus, it avoids "this bottleneck by exploring the relationships between items first, rather than the relationships between users" [235, p. 285].

Slope One Algorithm

In 2005, Yahoo published an algorithm [175] that will be transferred to the educational domain in this dissertation:

[16] An introduction to and explanation of the user–item matrix is given in Appendix B.11.

[17] This extract originated from a paper on a social network recommender that uses both Item-based as well as User-based Collaborative Filtering and was written primarily by Christopher Krauss.

"The Slope One algorithm (Item-based Filtering) calculates the Top-N items by taking into account the ratings [or other kinds of feedback scores] of all other users. It is divided into two parts (cf. [278, p. 153]). First of all there is a method to get the average deviation of two items:

$$dev(i_1, i_2) = \frac{\sum_{u \in U_{i_1 i_2}} (r_{u,i_1} - r_{u,i_2})}{|U_{i_1 i_2}|} \tag{2.6}$$

Where i_1 and i_2 are the items, r_{u,i_1} is the item's rating user u gave. $U_{i_1 i_2}$ is the set of users who rated both items and $|U_{i_1 i_2}|$ is its cardinality. So the result is the ratings deviation of an item. If the average rating of item i_1 is higher than the average rating of item i_2 the value is positive, if it is lower the value is negative, or if the average ratings are equal the value is $dev(i_1, i_2) = 0$"

Cf. [167, p. 66][18].

Figure 2.4 illustrates the two calculation steps of the algorithm: The left side represents the determination of the deviation in step one. In this case, the relevance score of item i_2 is determined for user u_2. According to step one, all neighbors are checked for their average rating deviation from i_2. The second step aims at predicting the missing rating by applying the determined rating deviation of others.

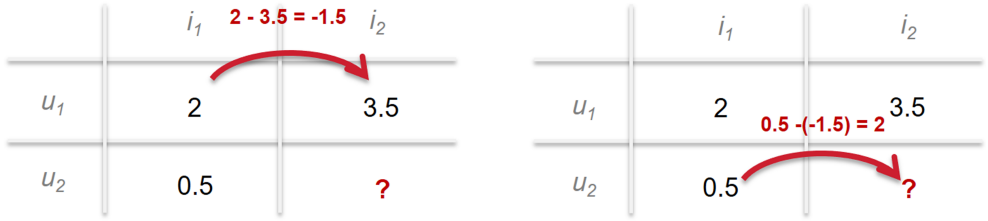

Figure 2.4.: Slope One Algorithm: Step one on the left side, step two on the right side.

"The prediction value $pre(u, j)$ for user u and item j is defined as

$$pre(u, j) = \frac{\sum_{i \in Ij} (r_{u,i} - dev(i, j))}{|Ij|} \tag{2.7}$$

Ij is the set of all relevant items to be compared with item j and $|Ij|$ is its cardinality. The higher $|U_{i_1 i_2}|$, the better the prediction. $r_{u,i}$ is the rating of user u for item i. (cf. [175, p. 3]) The resulting value is the predicted rating of this user u for the current item j.

The result will be limited to the specific rating interval, just in case, the predicted rating is not in that range. This might happen when the predicted rating is not in the range of [1,10],

[18] The implementation of Slope One represents a key algorithm in the publication of the TV Predictor *Recommender System* and was implemented by Christopher Krauss. The paper extraction was also written by him and reflects the work of [175] and [278].

> for instance when the current user always rated the items worse than the average user (e.g., a standard deviation of 3.5) and the requested item is rated extremely bad (e.g., 2.1). The predicted rating will be -1.4, so it must be mapped to the original range – in this case to 1"
>
> Cf. [167, p. 67][19].

The right side of Figure 2.4 shows the transfer of the deviation calculated in step one to the requested user–item tuple. Slope One is easy to implement, but a robust algorithm to predict unknown relevance scores (such as ratings or item accessed). The accuracy of the 2013 implementation of the Slope One algorithm in the TV Predictor[20] is 80.2% [167, p. 69]. However, a lot of improvements have made the algorithm even more accurate. These improvements include the splitting in a good and a bad rating dataset for a bipolar Slope One [175], uncertain neighbor optimization [177] or the inclusion of time weights [142].

Association Rules

An alternative *Item-based Collaborative Filtering* algorithm, which builds the basis for the generation of learning paths in this work, is known from Amazon's product recommendations: "Users who bought product A also bought product B and C". The underlying Association Rules were also used for program predictions in the TV Predictor engine.

> "Association Rules have the goal to find highly represented relations (so-called transactions) between a user or even a user's attribute and the items or their attributes (cf. [180, p. 441], [182, p. 500]). The resulting set of frequent items may, in addition, be scanned for some rules and afterwards the list of transaction can be divided in causations and consequence. For instance when a defined number of users have watched the same programs, the resulting frequent item set could be 'Program 1, program 2 and program 3 are often watched together.' and moreover if a user has watched a subset of these transactions, an association rule could be: 'You watched program 1 and program 2, but you may also like program 3.' and can be written as $\{Program1, Program2\} \implies \{Program3\}$ or more generally
>
> $$X \implies Y \tag{2.8}$$
>
> The item set X implies the item set Y (cf. [118, p. 1455])"
>
> Cf. [167, p. 67][21].

Transfer to TEL

As shown above, *Item-based Collaborative Filtering* can be very powerful, but its performance

[19] The paper extraction was written by Christopher Krauss and reflects the work of [175] and [278].

[20] The TV Predictor is a hybrid *Recommender System* developed by Christopher Krauss as employee of the *Fraunhofer-Institute for Open Communication Systems*. It was developed as prototype of his master thesis, afterwards extended to become a showcase for RTV (a Bertelsmann Arvato company) to be displayed on SmartTVs (shown at IFA 2012) and afterward realized as website component "Movisto" for `www.rtv.de` (Dec 2012 until June 2013).

[21] The paper extraction was written by Christopher Krauss and reflects the work of [180], [182] and [118]. Details of the algorithm, as well as some interesting mined rules on features in a German TV experiment, are presented in the Appendix (see Section B.12).

slows down with an increasing number of users and items. Within this dissertation, some ideas regarding Association Rules are transferred to the application area of *Technology Enhanced Learning*: Based on a list of transactions, a set of frequent, but here sequential, rules are determined. These learning paths represent the sequence of the next *Learning Objects* which are essential to consume to reach the course goal.

2.2.2.2. User-based Filtering

User-based Collaborative Filtering focuses first on finding similar users, the so-called neighbors. The prediction step, in turn, takes only the ratings of the nearest neighbors into account. Thus, most algorithms can save calculation steps, as they do not take the whole user–item matrix into account. The result is a Top-N recommendation list of items that are very popular or frequently consumed by similar users [80, p. 550].

Clustering is a *Machine Learning* approach that groups elements by the similarity of their attributes and can be used to group users according to their behaviors. Clustering

> "Cluster Analysis is also called Data Segmentation and has the goal to divide a set into subsets. It aims at finding groups of elements that are similar in one or more criteria (cf. [122, p. 454]). Amazon, for instance, uses a not further named greedy cluster generation, which starts with a set of randomly chosen users and searches for their nearest neighbors. Some of their algorithms classify the users into multiple clusters depending on the users' behaviors. (cf. [180, p. 77]) The TV Predictor uses K-Means (cf. [151, p. 675]), a partitional Clustering algorithm, in order to group users by respecting their likings on different attributes, such as preferred genre, category, and channel. Partitional means that the resulting clusters consist of [a] predefined amount of 15 subsets"
>
> Cf. [167, p. 67][22].

Besides Cluster Analysis, the *K-Nearest-Neighbour (KNN)* approach also identifies similar users[23]. The difference is that *KNN* determines a fixed number of similar users to a particular one and K-Means Clustering generates a fixed number of user groups – each with a variable number of users. Both, Cluster Analysis as well as *KNN*, are used in this dissertation to identify similar learners based on their studying activities.

2.2.2.3. Challenges for Collaborative Filtering

The vast amount of collected user and item data brings some challenges[24]: "In any recommender Sparsity Problem

[22] The section about K-Means clustering was realized and written by Christopher Krauss.

[23] An alternative approach to determine the similarity of users is to use the Pearson Correlation Coefficient. As it plays a minor role in the context of this work, it is introduced only in Appendix B.13.

system, the number of ratings already obtained is usually very small compared to the number of ratings that need to be predicted" [9, p. 740]. The consequence is a lack of the required relations between users and items[25]. The sparsity of the user–item matrices brings trouble to a *RS* in terms of performance and accuracy issues. The lower the amount of rating data that exists for specific users, the worse is their pattern predictability. And the lower the amount of rating data for a particular item, the lower the chance of its being recommended [302].

> "Jia Zhou and Tiejian Luo are grouping the approaches [to overcome sparsity] into two general classes [303]:
>
> - Dimensionality Reduction: Less important rows and columns of the matrix are ignored. This implicates a loss of possibly useful information.
> - Find and Add Additional Information: By means of the similarity measure rows and columns are filled with values"
>
> Cf. [163, p. 366][26].

This problem comes in different flavors: A new service shows a lack of user and item data. An established service, in contrast, might show only missing data, when new users register to the service or new items are added. The following sections focus on the main subtopics of the sparsity problem.

Cold Start Problem

The Cold Start Problem [302], [102] "occurs when a new user or item has just entered the system, it is difficult to find similar ones because there is not enough information (in some literature, the cold start problem is also called the new user problem or new item problem" [253]. This problem is especially present for the closed-corpus recommendations in the *TEL* domain where newly enrolled learners start to study the course materials at the beginning of a course. The main goal is to provoke new users into providing any significant feedback in order to obtain some information about some of their habits. Therefore, most services present a set of items, right after the registration process and before starting the actual service. Lam et al. use "collaborative and also content data to address cold-start" in a cold-start recommender [171]. Other services show the most frequently rated items in the system [167], some concentrate on the most meaningful items (e.g., the most controversial items) [152] and others again show only random items to obtain data on niche items. In closed-corpus settings, non-personalized item recommendations should refer to the didactic structure given by the teachers.

[24] Besides the issues mentioned in this section, Appendix B.14 lists more challenges for *CF*, such as the "New Item" problem, the "Back, Grey and White Sheep" problem and "Item Limitations".

[25] Appendix B.15 contains an analysis of the user–item matrices of popular web services and their calculated sparsity.

[26] This paper excerpt was written exclusively by Christopher Krauss.

2.2.3. Knowledge-based Filtering

Knowledge-based Filtering

Another (sub-)class of *Recommender Systems* is *Knowledge-based Filtering (KbF)* which "uses knowledge about users and products to pursue a knowledge-based approach to generating a recommendation, reasoning about what products meet the user's requirements" [51, p. 180]. For *Technology Enhanced Learning*, Manouselis et al. [191, p. 398] differentiate between *Case-based Reasoning*, which learns attributes of items often consumed by the user, and "Attribute-based techniques", which rely on explicitly entered attributes and that might also include non-item attributes. According to Burke [52], these filter systems do not require rating data – he assumes that *Content-based Filtering* requires feedback data from users on items. However, knowledge-based systems started as a subclass of *Case-based Reasoning (CBR)* and rely on user inputs regarding certain features that can subsequently be associated with item metadata. Users define their preferences for general item features directly in their user profile. Thus, *KbF* develops "knowledge about how a particular item meets a particular user need, and can, therefore, reason about the relationship between a need and a possible recommendation" [52, p. 3]. For instance, learners tell the system that they are beginners in specific topic areas, and the system starts filtering appropriate items based on this information. While Burke sees *Knowledge-based Filtering* as the "third type of recommender system[s]" [51, p. 180], Manouselis et al. [191, p. 398] are convinced that such an attribute-based system is preferably a sub-class of *Content-based Filtering*. This work follows the proposition of Burke[27], even though *CbF* also relies on the similarity of item features (without the necessity for rating data), such as for Item-to-Item Similarity Mapping [181]. However, the actual classification might depend on the main characteristics of the algorithm, which can also be closer to the characteristics of attribute-based systems, since a *KbF* approach can implicitly develop a knowledge-based user profile by consequently training item features as an alternative to user preferences. *Knowledge-based Filtering* will play an important role in this work as it is the class of a self-devised algorithm, called *Smart Learning Recommender*.

2.2.4. Hybrid Filtering

Hybrid Recommendations

Although each introduced filtering approach is very accurate in its core domain, a typical web service offers more than just one recommendation type. For instance, Amazon names the category of recommendations on their home screen "Recommended for you". In contrast, recommendations on a product page are called "Frequently Bought Together", "Customers Who Bought This Item Also Bought", "What Other Items Do Customers Buy After Viewing This Item?" or "Sponsored Products Related To This Item". In total, Amazon shows seven different recommendation types

[27] In the book "Recommender Systems for Learning", Manouselis et al. introduces *KbF* also as an additional recommender class by following Burkes definition – besides *Demographic Filtering* and *Utility-based Filtering* [190].

alone on a Blu-Ray product page [28]. The inclusion of different recommendation techniques is called *Hybrid Filtering (HF)*.

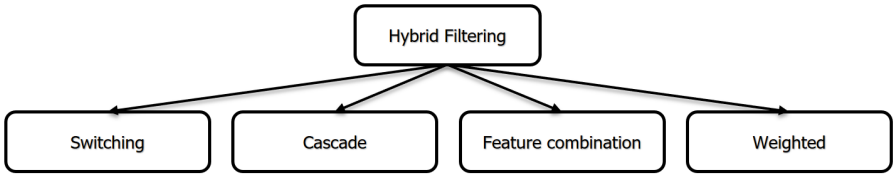

Figure 2.5.: Sub-classes of hybrid filtering in *Recommender Systems*.

Hybrid Filtering Classes

Depending on the request type *HF* algorithms can be grouped into the following classes (see Figure 2.5):

1. Switching Filtering uses just one available approach.
2. Cascade Filtering sequentially uses different algorithms. Thus, each approach constrains the resulting set to avoid unnecessary calculations on already rejected items.
3. Feature Combination transfers algorithms from a specific filtering approach to work in another filtering context.
4. And finally Weighted Filtering merges the single results by weighting them.

Weighted Filtering

The weighted filtering approach is the basis for the *Smart Learning Recommender*, which is a core algorithm of this work and requires a special formula for the combination of the single results:

$$RV = \frac{\sum_{i=1}^{N} w_i * r_i}{\sum_{i=1}^{N} w_i}. \tag{2.9}$$

The recommendation value RV results from the summation of each single recommendation value r multiplied by the according weight w of the N different algorithms and afterwards divided by the summation of all weights.

2.3. Conclusions for Recommender Systems

As shown above, there are three major classes for filtering items – namely *Content-based Filtering*, *Collaborative Filtering* and *Hybrid Filtering* – each having its own characteristics, techniques and challenges. One additional approach, *Knowledge-based Filtering*, is of particular interest to this work. As a conclusion, the findings of this chapter and their possible impact for the specialized area of *TEL Recommender Systems* are formulated in the following hypothesis.

Scientific Hypothesis /SH1.0/:

***Recommender Systems* that predict appropriate recommendations for course items may rely on similar techniques to traditional *Recommender Systems*. The basis is**

[28] See Amazon: `www.amazon.com` (Accessed: 27.08.2016)

user and item metadata and its algorithms aim at predicting relevance scores in order to generate Top-N lists. Educational approaches face similar issues, such as cold start and sparsity, and the algorithms can be classified according to the general *Recommender System* taxonomy.

The next chapter discusses related works in the area of *Recommender Systems* that have been applied in *Technology Enhanced Learning*. Thereby, the characteristics of a particular paradigm for educational *RSs* are determined.

3. Related Work on Recommender Systems for Technology Enhanced Learning

Today, learners frequently do both: learn formally (e.g., explicitly in classrooms) as well as informally (e.g., implicitly at work)[29]. This dissertation aims at supporting a variety of educational settings, especially Online Courses [115, 68], Blended Learning Courses (which add complementary digital content to a regular classroom setting [15]) and digitally supported Face-to-Face lectures[30]. These settings represent formal or partly formal learning settings. Learning environments which are supported by software and/or hardware components are summarized as *Technology Enhanced Learning (TEL)*. *TEL* "generally covers technologies to support teaching and learning activities, including recommendation technologies that facilitate retrieval of relevant learning resources" [270, p. 319].

About this Chapter

While classical *Recommender Systems* try to collect as much information as possible for the prediction user preferences – and perform very well for traditional application areas such as e-commerce or media services – educational environments follow a particular paradigm. Manouselis et al. note that "information retrieval goals that *TEL* recommenders try to achieve are often different to the ones identified in other systems" [191, p. 388]. This chapter introduces the area of *Recommender Systems* in *Technology Enhanced Learning (TEL)* that enables adaptability of learning platforms to the learner's needs. Moreover, this chapter demonstrates that *TEL Recommender Systems* should be context-aware and time dependent. That is the reason why two additional *Recommender System* classes are introduced in this chapter: *Context-Aware Recommender Systems* and *Time-Aware Recommender Systems*.

3.1. Personal Learning Recommendations

A personalized, learner-centric platform in the area of *TEL* is also called *Personal Learning Environment (PLE)* [58]. In contrast to typical *Learning Management Systems (LMSs)*, *PLEs*

[29] An introduction of recent formal and informal learning concepts as well as Online Courses and *Massive Open Online Courses* is presented in Appendix B.16.

[30] There are also other popular course settings, which might be based on different didactic concepts, such as Flipped Classroom Learning [157, 85], or on a technological basis, such as *Massive Open Online Courses (MOOCs)* [83, 258, 296]. While these will not be part of the evaluation, the introduced *RS* algorithms might also support these environments.

create a "knowledge-pull" environment that meets the individual "needs from a wide range of high-value knowledge sources" [59, 58]. *Recommender Systems* help to personalize learning environments through individual content suggestions that fit the learner's needs. Thus, *RSs* help to identify suitable items for learning and teaching tasks from a wide range of available learning resources [269, 58]. These recommended items are typically *Learning Objects* within a course, which is also the main application area for this work. However, a recommendation can also refer to other courses offered on the same platform [191] or present other recommended elements, such as appropriate classmates for group learning [221] or external material in the *World Wide Web (WWW)* for further informal learning – these are called open-corpus recommendations [48]. An analysis of published research by Verbert et al. [270, p. 326] indicates that the most recommended entities in the area of *TEL* are: 1) "learning resources" with a coverage of 100%, "peer helpers", i.e., other learners, with a coverage of 32%, 3) "messages/ hints/ triggers" at 23% and finally 4) other courses at 5%.

Learning Objects

According to Polsani [218] there are multiple definitions for *Learning Objects*. For instance, an *LO* can be "the smallest independent structural experience that contains an objective, a learning activity and an assessment" [218][31]. In another definition "a *Learning Object* is defined as any entity, digital or non-digital, that may be used for learning, education or training" [218][32]. Thereby, "*Learning Objects* are a new way of thinking about learning content. Traditionally, content comes in a several hour chunk. *Learning Objects* are much smaller units of learning, typically ranging from 2 minutes to 15 minutes" [218][33]. This work follows the first definition (of L'Allier who defines an *LO* as the smallest independent learning element). This is sometimes also called a Learning Nugget. From a *Recommender System* point of view, a *Learning Object* is an "item" which can receive feedback from the learners and which can be presented as a recommendation.

PLE Examples

In the following, two representative examples of a *Personal Learning Environments* are introduced. The first one is offered by the industrial learning technology provider IM-C[34]. In order to maintain the learner's motivation at a high level, IM-C introduced individual experience points: a Gamification element, which represents the course goal (a pre-defined number of points which are needed to reach the course goals) as well as the learner's progress within the course (user's earn points by performing subtasks). Thereby, learners earn most experience points by completing formal learning tasks, and a smaller percentage of the total points is earned through informal

[31] The work of Polsani [218] is a secondary source. The primary source is "L'Allier, James J. (1997) Frame of Reference: NETg's Map to the Products, Their Structure and Core Beliefs. NetG." `http://www.netg.com/research/whitepapers/frameref.asp` However, this page is not available anymore.

[32] The work of Polsani [218] is a secondary source. The primary source is "IEEE Learning Technology Standards Committee (LTSC) (2001) Draft Standard for Learning Object Metadata Version 6.1." `http://ltsc.ieee.org/doc/`. However, also this information is not available anymore.

[33] The work of Polsani [218] is a secondary source. The primary source is "IEEE Learning Technology Standards Committee (LTSC) (2001) Draft Standard for Learning Object Metadata Version 6.1." `http://www.wisc-online.com/Info/FIPSE%20-%20What%20is%20a%20Learning%20Object.htm`. This information is also not available anymore.

[34] IMC. See: `https://www.im-c.com/` (Accessed: 13.04.2017).

learning tasks. The particular distribution of formal and informal points, as well as the specific informal tasks, e.g., watching a related video on Youtube, are personalized through the suggestions of a *Recommender System*. Surveys at the beginning of a course help to classify the user and, in turn, predict an individual experience path for each learner [35].

The second example, PAL3, was developed by Swartout et al. [254] and uses Gamification and skill level observation. A digital avatar gives hints and recommendations for learning. Thereby, Swartout et al. identified four mechanisms to increase engagement:

1. "Flow: Promoting interaction and flow by presenting a steady stream of short, varied learning activities.
2. Gamified Learner Models: Presenting progress and loss-of-progress (forgetting) with open learner models.
3. Social Motivation: Encouraging social use and competition, such as through leaderboards.
4. Accumulated Rewards: Progress-based system expansion (e.g., unlocking content and customizations)" [254].

The PAL3 system supports both: "adaptive recommendations [within a course] and open learner models that are intuitive to students and instructors" [254]. The latter abstracts high level skills to enable *Life-Long Learning*.

3.2. The Special Educational Recommendation Paradigm

Educational *Recommender Systems* follow a particular paradigm that is different from the e-commerce or entertainment domain. *TEL RSs* need to consider the special characteristics of educational settings comprising the learners' motivation, the learners' needs, typical user behaviors and resulting activity data. This section introduces the main parameters of this paradigm. The following sections, in turn, analyze the related work of *Collaborative Filtering*, *Context-Aware Recommender Systems*, *Time-Aware Recommender Systems* and Learning Paths with a particular focus on the determined characteristics.

Learning & Teaching Styles

Felder and Silverman [107] observed different learning and teaching styles in an engineering education setting. The resulting list of four didactic dimensions[36] presents an approach for learning style classifications that is widely accepted as a universal model for learning and teaching. Thereby, each learner and teacher can be graded in the four dimensions between "active & reflective", "visual & verbal", "sensing & intuitive" and "sequential & global" [107, p. 675]. These learning styles allow for a better understanding of typical learning and studying patterns. A "verbal"

[35] IMC – Experience Points for innovative learning. See: `https://www.im-c.com/sg/learning-technologies/innovation-pack-7/`. (Accessed: 13.04.2017).

[36] Actually, the Felder-Silverman Model comprises five dimensions, but Richard M. Felder later reduced it by the inductive–deductive dimension [108]. Honey and Mumford devised a similar model [128].

student, for instance, should receive less "visual" content recommendations, while a "sequential" learner might not want to deviate from the didactic content structure.

Characteristics

In *TEL Recommender Systems*, "each learning setting has its own characteristic data sets, user models, recommendation tasks, and most suitable recommendation algorithm" [92, p. 2850]. Buder and Schwind [50] explicitly identified the need for adaptations of common commercial *Recommender Systems* in order for them to be appropriate for learning environments. This is why most researchers borrow traditional *Content-based Filtering* and *Collaborative Filtering* algorithms and, at least partially, adapt them to the domain of *Technology Enhanced Learning* instead of directly applying the same approaches [58]. While some educational *RSs* utilize *Hybrid Filtering* algorithms [110, 267, 150], most can be classified into *CF* and *CbF*.

Bauman and Tuzhilin [29] categorized educational *Recommender Systems* either as "knowledge enhancing" which aims at broadening the knowledge to new topics or as "remedial" which aims at filling identified knowledge gaps of previously studied items. Closed-course *RSs*, such as the ones introduced in this work, aim at solving both issues. They assist the learners in reaching their personal or course-specific goals which require new topics to be learned and the consolidation of already acquired knowledge.

Learning Goals

Learning goals may be defined by different stakeholders and levels – e.g., by individual learners, by educators and content creators, the institutional staff and even on the political level (see Ifenthaler and Schumacher's layered model of educational stakeholders [135, p. 177]). This work mainly focuses on learning goals given by educators, in terms of formal defined learning objectives and assessments. The goals of a learner might be similar to those of the teacher – for example when a student only wants to pass the final exam. However, learners might be interested in only some of the offered topics or participate in the course for another reason than actually expected by the educator. Without requesting information about the individual learning goals, it would be tough to recommend appropriate materials. Thus, the main aim of the closed-course *Recommender Systems* in this work is to support an effective and efficient knowledge acquisition of the course topics. At the same time, the *Recommender Systems* aim at enabling learners to pass the course with the highest possible individual results (for example, measured in assessment tests).

TEL Recommender

Verbert et al. identified an important research question: "how [can] generic algorithms [...] be modified in order to support learners and teachers" [269, p. 16]. The fundamental differences between traditional and learner-oriented *Recommender Systems* investigated in this work are summarized in the following list and then described in the subsequent sections.

- A *TEL Recommender System* should make learning more efficient and effective for its users.
- Educational recommendations should (positively) influence future learning activities.
- Instead of preferences, a *TEL RS* should be based on the learning needs of its users.
- Closed-course systems collect more activity data which lead to a fast increasing user–item

matrix density. This must be taken into account by a *TEL Recommender System*.

- A *TEL Recommender System* should incorporate multi-dimensional contextual information.
- Activity data in closed-courses show a high dependency on time, which should be taken into account by a *Recommender System*.
- A *TEL Recommender System* should respect the interdependency of its items.

3.2.1. Learning Efficiency and Effectiveness

Learning, especially *Self-Regulated Learning*, is a time consuming task that requires responsibility on the part of the learner. This can be assisted by a *Recommender System*. **An educational closed-course *Recommender System* aims at making learning more efficient and more effective.** Thereby, "efficiency" describes the way to achieve a goal in terms of efforts, process and time. "Effectiveness", in turn, directly concerns the result achieved. In terms of learning in a closed-course setting, a higher efficiency means optimizing the process, saving effort and time to reach the same course goal. A higher effectiveness means to reach a higher goal, e.g., a better mark in the exam or longer lasting knowledge. Both can be improved through the decision support of *Recommender Systems*. Nava Tintarev et al. [264] present the following criteria for good recommendations:

- Transparency: Explain how the system works.
- Scrutability: Allow users to tell the system it is wrong.
- Trust: Increase users' confidence in the system.
- Satisfaction: Increase the ease of usability or enjoyment.
- Persuasiveness: Convince users to try or buy – or in the case of *Technology Enhanced Learning*, to motivate the user to learn.
- Efficiency: Help users make decisions faster.
- Effectiveness: Help users make good decisions.

An educational *RS* must respect all criteria. However, efficiency and effectiveness of *Learning Object* recommendations are of particular interest in *Technology Enhanced Learning*, because the two criteria directly affect learning and teaching styles [220]. Both are core aspects in this dissertation and refer to "appropriate" recommendations.

3.2.2. TEL Recommendations Influence Future Activity Patterns

Manouselis et al. describe studying, learning and recommending *Learning Objects* for *Technology Enhanced Learning*, as "an effort that takes more time and interactions compared to a commercial transaction. Learners rarely achieve a final end state after a fixed time. Instead of buying a product and then owning it, learners achieve different levels of competencies that have various

levels in different domains" [191, p. 391]. According to these authors, learners find appropriate contents for the preparation of a lesson in order to:

1. Become motivated.
2. Recall existing knowledge.
3. Illustrate, visualize and represent new concepts and information .

In contrast to traditional *RSs*, educational *Recommender Systems* try to actively change the user's future activities. A movie recommender aims at predicting the users' future activities based on their current interest in specific aspects – e.g., high rated genres. This information on their interests is then extrapolated to predict the users' next interests in new items. Implicitly, a traditional *Recommender System* trains past user patterns to predict future item consumption. **An educational *Recommender System* must additionally fulfill pedagogical tasks, such as user motivation and support to reach the course goals.** Weak learners, for instance, should not receive recommendations based on their current learning patterns but rather on successful patterns. Vygotsky notes that "recommended learning objects should have a level slightly above learners' current competence" [277] (found in [191, p. 391]). Thus, a challenge for an educational *Recommender System* is to determine successful learning patterns.

3.2.3. Relevance Score: Learning Need Instead of Preference Value

The learner's profile in a learning recommender can show enormous discrepancies compared to traditional user profiles due to the nature of the learning process "which is closely connected to educational, psychological, social, and cognitive science" [94, p. 7]. One characteristic comes from the divergent user motivation. Educational systems generate another kind of interaction data because of their specialized content offering [223].

Knowledge

An important difference between traditional *Recommender Systems* and educational *Recommender Systems* lies in the type of the user–item relation. **Closed-course *RSs* must consider the users' knowledge instead of their preferences.** The values in the user–item matrix rely on the type of input data and are consequently used as the recommender's relevance score. Of course, preferences help to motivate learners, but to reach a given course goal, users must study all relevant items irrespective of the items' popularity. Schatten and Schmidt-Thieme noted that a typical task of a *TEL Recommender System* is to retrieve "the knowledge of the student from the given information, e.g. score, time needed, previous exercises, etc." [237]. However, these numeric data are "just an indirect representation of the knowledge, which cannot be automatically measured, but needs to be modeled inside the system" [237].

Knowledge Modeling

The learner's model comprises representations of the actual knowledge $ActualKnowledge_{u,i}$ of learner u about a *Learning Object* i at a particular point in time t. This knowledge can be presented on a scale from 0% (no knowledge) up to 100% (highest possible knowledge). Experts

must define the latter, which might be the required knowledge to reach the best mark in a test. The $ExpectedKnowledge_i$ is also the reference for the highest possible knowledge of item i. The deviation between the actual knowledge and the expected knowledge can be expressed through the knowledge gap:

$$KnowledgeGap_{u,i,t} = ExpectedKnowledge_i - ActualKnowledge_{u,i,t} \tag{3.1}$$

Learning Need

It is of interest for the learner to obtain suggestions regarding missing knowledge in order to compensate for these gaps by learning [94, p. 9]. Besides the $KnowledgeGap_{u,i,t}$ which can be approximated by algorithms [237], educational *Recommender Systems* might rely on other relevance scores, as well. For example, the number of *LO* accesses of similar learners or the timely relevance of a topic presented in the presence lecture. **"Learning Need" is defined in this work as the representation of the necessity of learning a particular item at a particular point in time.** The Learning Need can consist of different factors, including the identified knowledge gap [103, 29]. Both values, preference value and Learning Need, are sub-classes of a recommender's relevance score and relate users to items.

3.2.4. Changing Knowledge-Levels and Increasing Matrix Density

The relevance scores are expected to change more frequently in educational *RS* because the learner's knowledge is expected to increase with every item interaction. Traditional *Recommender Systems* try to model a user profile, which becomes more stable with every item consumption. With a sufficient amount of user data, a new rating of a movie is expected to reflect the recommender's initial prediction of this particular rating. If the rating differs from the prediction, the movie recommender slowly adapts to the new preferences – because the user's interests are expected to shift only slowly. However, it is unlikely that the users' preference will change to the opposite in short time, especially for all items of the huge item set within a closed-corpus system.

In *RS* for *TEL*, less items are offered than in e-commerce systems [94]. User profiles change frequently in *Technology Enhanced Learning* [19, p. 138 - 148] [103, p. 327]. Knowledge levels, and so also the related learning need values, must be adjusted after every learning iteration. At best, a learner starts without any knowledge and acquires the highest possible knowledge by the end of the course.

A *TEL Recommender System* collects more data within a short time period compared to traditional *Recommender Systems*. Thereby, a course starts with no, or only little, information on the learners' previous knowledge. However, during the period of the course, the learners are expected to study all relevant items or at least provide some feedback on the items [155, 109]. They can, for instance, provide feedback when they assume that they already have

expert knowledge and, consequently, skip those items. Particularly challenging is the fact that all learners start studying at the same time. Thus, the cold start problem applies for all users simultaneously.

> "Most *Recommender Systems* need to handle sparse data, which is a big challenge for *Collaborative Filtering*. At the beginning of a course, the user-item-time-matrix is sparse as well, because [the recommender] knows very less about the knowledge of a user. However, over the period of a course and with increasing user interactions, the matrix becomes dense. Thus, [the educational recommender] needs to be capable of processing both: sparse user data during cold-start and rich user data at the end of course"
>
> Cf. [162][37].

A traditional *RS* rarely processes dense datasets [303], but, especially at the end of a course, almost all (active) learners should have consumed all relevant learning items. A hybrid approach, consisting of *Collaborative Filtering* and *Collaborative Filtering*, can compensate for the drawbacks of single methods and phases at different points in time by, for instance, weighting the results of various algorithms according to the density of the user–item matrix.

3.2.5. Multi-Dimensional Contextual Information

Approximating and predicting the learner's knowledge requires more contextual data compared to approximating ratings. While ratings express the users' opinions on items, the learners' self-assessments (which are the equivalent) regarding their knowledge of particular topics are likely to be inaccurate. Learners are often not aware of the actual learning requirements and the scope of the assessments. Thus, it is probable that they overestimate or underestimate their knowledge. Moreover, a reliable value of their actual knowledge is often not generated until the final assessment at the course end. That means, a *Recommender System* must rely on other data in the period before the assessment. Researchers determined that a set of multiple attributes are appropriate to estimate the learners' knowledge levels [191, 58, 96]. This comprises various information on the users, the items, the context and the time [293, 213].

3.2.6. Time-Dependency

Drachsler et al. note that "learners and LAs [(learning activities)] change over time" [94, p. 6] which also directly affects the learner's goal (also see [103, p. 327]). **Educational recommendations depend more on time information than do traditional recommendations.** In most traditional *Recommender Systems*, only the latest feedback of a user regarding an item is considered for multiple reasons:

[37] This excerpt was exclusively written and presented by Christopher Krauss at a doctoral consortium.

1. Users usually consume the same item only once or with significant pauses between the repetitions.
2. Often, the point in time of an item consumption plays a minor role – or is even not determinable, as with movie rating systems such as Movielens [120], where the time of rating is not necessarily the same as the item consumption time.
3. In general, users are not expected to frequently change their opinion on the same item.
4. The last user feedback is of particular interest as it represents the user's most recent opinion on an item.
5. Existing two-dimensional prediction models work well on the known user–item relation for traditional *Recommender Systems*.

Timely Relevance

In *TEL*, the relevance score is expected to change after each learning process. Thus, following a recommendation of a *Learning Object* has a direct effect on the items relevance score, because the user's knowledge increases with each learning repetition. The consequence is that educational *Recommender Systems* need to incorporate also the historical consumption and feedback data from different points in time in order to transfer learning patterns. Thereby, educational recommendations show a high degree of time dependency, because these recommendations generally correlate with the learners' knowledge levels and the progress of the course. Learning happens in different stages and phases that directly affect its efficiency and effectiveness [98, 246, 23]. Also courses run through different phases, e.g., starting with the enrollment and cold start phase, lecture or assessment periods, holidays and breaks as well as learning phases before the final assessment.

In particular for *TEL*, certain recommendations become obsolete after a short time span, e.g., when the *Learning Object* has been studied by the students or the next lecture focuses on a different topic. This leads to the requirement of recommending and measuring timely relevant items which must be taken into account for the prediction of appropriate *Learning Objects*.

When considering historical data, the classical user–item matrix becomes insufficient as it just allows one feedback value per user–item pair to be stored. *Time-Aware Recommender System* extend the traditional user–item matrix by a time dimension.

3.2.7. Interdependency of Items

Items in *TEL Recommender Systems* might have dependencies on other items of the same course [244, 21, 271, 94] – more than, for example, movies may depend on each other. An *LO* might introduce the basics that are needed to understand a subsequent *Learning Object* which is why the learner should first focus the basics before studying the second item. Another *LO*, in turn, might deepen the knowledge of a previous topic and so on. However, two course items can also contain separate topics that may be learned independently from each other. Thus, **educational *Recommender Systems* should take given structural relations between different items**

into account.

3.3. TEL Recommender Systems Utilizing Collaborative Filtering

After the introduction of the special paradigm for educational *Recommender Systems*, the following sections analyze related work with particular respect to the determined characteristics of *Technology Enhanced Learning*. Thereby, *Collaborative Filtering* is an often applied and at the same time promising approach (cf. [191, 54]).

> "Manouselis et al. argue that more than the half of all published *Recommender Systems* in the area of *Intelligent Learning Technologies* were still at a prototyping or concept level and only 10 have been evaluated in trials with real participants [191]. Most of these systems are designed to predict items in a closed system using the two-dimensional *Collaborative Filtering* user-item-matrix"
>
> Cf. [161, p. 501][38].

In 2015, Erdt et al. [103] published a survey which analyzes 235 articles about *TEL Recommender Systems*[39]. The survey indicates that *Recommender Systems* in *TEL* gain increasing attention in the last few years. While only 18 related papers have been published between 2000 and 2005, 160 have been published between 2010 and 2015. However, only 13% (30) of all algorithms and systems have been tested in real life settings and even 42% had no evaluation at all [103, p. 331-336].

Closed Corpus

Collaborative Filtering is a frequently used approach to educational recommendations. The core assumption is that learners show similar learning activities for the same items within a course and thus their learning patterns can be processed to predict future *Learning Objects*. Obviously, classical *CF*-approaches are the first choice to identify appropriate items in *Technology Enhanced Learning* systems. Examples are:

> "'CourseRank' [160] [, an unofficial course recommender] of the Stanford University, 'Altered Vista' [224] that uses Association Rules of frequently used learning objects in courses or 'RACOFI', a rule-applying *Collaborative Filtering* system 'that assists on-line users in the rating and recommendation of audio (Learning) Objects' [42]. However, these recommenders only work on a flat item hierarchy and without time or extended context data. Nevertheless, it seems to be very important to include the intrinsic and extrinsic motivation of students,

[38] This paper excerpt was primarily written by Christopher Krauss. However, the study of Manouselis et al. was published in 2011.

[39] Note that multiple of the 235 papers describe the same *Recommender System* and, thus, the number of 235 papers does not reflect the amount of *Recommender Systems* which is expected to be much lower.

in terms of 'pedagogical aspects like prior knowledge, learning goals or study time' [191]"

Cf. [161, p. 501][40].

CF in *TEL* often utilizes ratings, as known from entertainment or e-commerce recommenders [170]. Thereby, the appropriateness of learning material for an individual user is not rated, but instead the learning object quality. The "ReMashed" platform of Drachsler et al. [95] gives access to different learning resources and recommends content based on preference ratings through the DUINE Recommender Framework [41]. Rafaeli et al. [221] present an approach which focuses on learning communities and offers either an automated *CF* algorithm or follows the real advice of friends to form suitable learner communities.

Thereby, utilization of an unspecific *Collaborative Filtering* approach is not necessarily practical. Sicilia et al. [247] experimented with standard algorithms, such as the Pearson Correlation Similarity and the Euclidean Distance, to recommend *Learning Objects* based on the behavior of similar users. In doing so, they received high error values which is an argument against the application of unmodified traditional *RS* approaches for *TEL*.

Open Corpus

Another promising approach to assisting learners is to additionally recommend items that are not part of the actual course [22, 259] – called open-corpus recommendations. Chatti et al. note that in open-corpus environments "learners rarely share the same or similar learning resources due to the fact that they follow their individual interests and preferences" [58]. Thus they suggest focusing more on Preference Filtering techniques by incorporating tagged user interests into traditional *Collaborative Filtering*. Therefore, Chatti et al. extend the user–item matrix by a new dimension that represents all user-generated tags, and this can be split into user–item, user–tag, and item–tag sub-matrices. Each sub-matrix, in turn, is used to calculate user- or item-based predictions with cosine similarity that are merged subsequently to a single Top-N list[42].

3.4. Context-Awareness in Recommender Systems

Most researchers conclude that *Collaborative Filtering* for educational *Recommender Systems* performs better when more than just one attribute type is incorporated. Manouselis et al. suggest the inclusion of context-aware aspects "to embed pedagogical reasoning into *Collaborative Filtering* driven recommendations" [191]. In addition, multi-criteria input on the items increase the suitability of the recommendations. These attribute dimensions comprise ratings of prior knowledge,

[40] The paper excerpt has primarily been written by Christopher Krauss.

[41] DUINE. See http://www.duineframework.org/ (Accessed: 13.04.2017)
Marginalia: The DUINE Framework is a generic *Collaborative Filtering RS* and was the first engine the author used for video recommendations in 2010.

[42] In 2016, Chatti stated that *Social Network Analysis (SNA)* is an appropriate approach for professional open-corpus *Recommender Systems* [60]. However, this dissertation focuses exclusively on a closed-corpus item catalog.

presentation styles and even attractiveness [191, p. 12]. That is the reason why a distinct area of *Recommender Systems* is introduced in this section – the *Context-Aware Recommender Systems*.

Pelanek et al. investigated the close correlation between the timing of problem-solving and multidimensional student skills that "may be useful for automatic problem selection and recommendation in *Intelligent Tutoring Systems* and for providing feedback to students, teachers, or authors of educational materials" [213].

The ISIS Recommender System of Drachsler et al. [93] processes the data of other learners as well as information on the users' activities and profiles for recommending *Learning Objects*. Especially interesting is the fact that they underline the significance of the attribute that represents the time needed to complete the *Learning Objects*.

Chatti et al. showed that user-generated tags seem to be useful in the *TEL* context, as they are familiar to the user and represent semantic values associated with an item that might not be an explicit part of the metadata [58]. Swertz et al. created a "Playground" to apply different pedagogical ontologies in adaptive e-learning environments, which can be utilized by researchers for evaluation purposes [256][43].

Similar to the work of Chatti et al. [58] and Manouselis et al. [191], the *Recommender System* "CoFind" analyzes multiple freely available repositories and thereby uses an "N-dimensional collaborative filtering" approach to detect *Learning Objects* fitting the user's preferences [96].

This work builds upon these findings (especially those of Manouselis et al. [191] where *Collaborative Filtering* requires user data of multiple attributes). *Recommender Systems* focusing on multi-dimensional contextual attributes are called *Context-Aware Recommender System (CARS)* and are frequently used for application areas apart from *Technology Enhanced Learning*[44].

3.4.1. Definition of Context

Yau and Joy note that adaptive learning environments should be context aware and, at best, adapt to the user's personal schedule and situation (e.g., regarding location, preferred learning style and available time) [293]. According to Dey, context is "any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves" [87, p. 4]. Moreover, Dey stated that a system is context aware "if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task" [87, p. 5].

Math. Description

Campos et al. [57] defined the context information of *RSs* as an extension of the user profile P_u associated with the user ratings of user u. However, in this work, r represents not only ratings,

[43] An analysis of additional educational ontology-based *Recommender System* is presented in Appendix B.17, but these play only a minor role in this dissertation.

[44] A brief overview of popular context-aware services is presented in Appendix B.18.

but a relevance score as defined in Section 2.1:

$$P_u = \{r_{u,i} | u \in U, i \in I, r_{u,i} \neq \emptyset\}. \tag{3.2}$$

This information can be associated with any user, item or experience – e.g., location, weather, device, mood or time [57, p. 72]. The inclusion of context information helps to increase the recommendation accuracy and thus the users' trust in the recommender as well as the level of sales [114, p. 671], [57]. Moreover, Cremonesi et al. note that the quality of a *Recommender System* might depend on the season. In their case, only in "the low season scenario the quality of [the] non-personalized algorithm is comparable to – or even better than – the quality of personalized algorithms" [71, p. 342] which underlines the necessity of taking context information into account.

Already in 2005, Adomavicius et al. [8] considered different contextual factors for the calculation of their recommendations. Their "multidimensional (MD) approach to recommender systems [...] can provide recommendations based on additional contextual information, [...] multiple dimensions, profiling information, and hierarchical aggregation of recommendations" [8]. Thereby, the traditional user–item matrix is extended by a set of context dimensions.

3.4.2. Context Dimensions

Later, Adomavicius and Tuzhilin [10, p. 224] defined a *Context-Aware Recommender System (CARS)* as extension of context to the *Recommender System* definition introduced in Section 2.1:

$$F : U \times I \times C \rightarrow R, \tag{3.3}$$

where C is a set of contextual information C_i of dimension i, that may have its own type and range of values. Moreover, each contextual dimension $C_1, C_2, ..., C_n \in C$ may have another representation of its data reflecting the complex nature of its information [10, p. 224], [57, p. 72]. Taking into account that U and I can also be represented as contextual dimensions (incorporating user and item data), the formula can be generalized as a Cartesian product over the space $C_1 \times ... \times C_n$ [10, p. 226]:

$$F : C_1 \times ... \times C_n \rightarrow R. \tag{3.4}$$

Filtering and Modeling

The method of incorporating additional contextual dimensions for the generation of recommendations can be manifold [10]. Nevertheless, three main concepts have been established for *Technology Enhanced Learning* [270, p. 328]: 1) "recommendation via context-driven querying and search" which is very similar to "contextual prefiltering" that is applied before using traditional recommender approaches, 2) "contextual post-filtering" that is, consequently applied to the outcome of traditional *Recommender Systems* and finally 3) "contextual modeling", as the most

advanced *CARS* approach, using algorithms for context data in the recommendation of items. The latter determines contextual dependencies of features which cannot be identified by simply filtering particular contextual features – an example is a recommendation of a specific item type in a particular situation (location and point in time) based on the previously trained patterns of other users.

Dimensional Classes

Different approaches exist to classify general contextual dimensions as proposed by Adomavicius and Tuzhilin [10]. For *Technology Enhanced Learning*, some adapted classes have been established, such as the differentiation between "who" (user), "what" (object), "how" (activities), "where" (location) and "when" (time) [178]. In this work, the definition of Verbert et al. [270, p. 322 - 324] is adapted to classify the dimensions as follows (an overview is presented in Figure 3.1):

1. A "User" is typically represented by a user profile and contains additional contextual sub-dimensions (such as interests, prior knowledge, learning goals or emotional status [88] – see Figure 3.1 for all sub-dimensions of the "User").
2. An "Item" is the resource that is typically accessed in learning environments and it also contains a set of sub-dimensions (some of which overlap with the sub-dimensions of the user, such as [required] knowledge, [media] type of learning and [intended] learning goals).
3. The "Activity" describes the type of action performed by a "User" typically on an "Item" which refers to the traditional user–item feedback in *Collaborative Filtering*.
4. "Time" can be a point in time or a time span and is commonly related to another context dimension, such as an "Activity". State changes over time are introduced in [239].
5. The "Location" can be represented as GPS coordinates or as labels, e.g., a classroom, the home or a particular city [239].
6. The "Computing Context" represents the characteristics of the used hardware, software and network (e.g., used devices, available bandwidth or the accuracy of GPS due to the signal strength [227]).
7. "Environmental Conditions" describe the relevant physics, such as light, noise or weather (e.g., temperature [47, 227] or more general weather information [8]).
8. "Social Relations" are associations between two or more persons or groups of people (e.g., friends and communities as well as information from social networks [239, 148, 186]). However, in a *TEL* system it can also refer to classmates and learning groups [190].

In reality, a lot of combinations and mixture forms appear that might not be attributed to only one context dimension. "People nearby the user" [239], for instance, incorporates the "User", "Location" and "Social Relations" dimensions and "Objects around" [239] comprises "User", "Location" and "Item" information.

The introduced dimensions enable analysis of all-encompassing activity patterns and the determination of the current situation with the prediction of future interactions. The *Smart Learning*

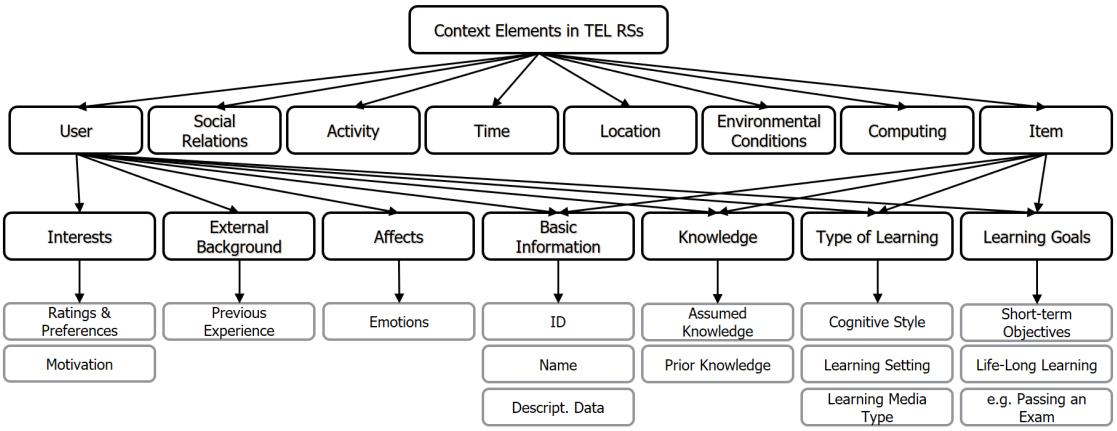

Figure 3.1.: A taxonomy of context elements for *Recommender Systems* in *Technology Enhanced Learning* (cf. [10] and [270, p. 322 - 324]); Black boxes represent the main dimensional classes and sub-dimensions of a "User" or an "Item"; Examples for the sub-dimensions are shown in the gray boxes.

Recommender which will be introduced in this work incorporates a set of these different contextual dimensions for closed-corpus recommendations in a *TEL* setting.

3.5. Time-Aware Recommender Systems

As stated above, the time dimension is of particular interest for *Technology Enhanced Learning*. That is the reason why this section introduces the distinct *RS* area of *Time-Aware Recommender Systems* which is actually not part of standard educational environments. However, this section discusses why it makes sense to incorporate time-information for *TEL Recommender Systems*.

The time dimension T allows for the modeling of transitions from a one-dimensional state to another – e.g., the interest towards an item at different points in time. Thereby, different data sources show varying dependencies with time – for instance, web search queries might either show periodic trends of "Activities" [272, 32] or temporal correlations of the users' "Interests" [65, 301].

Time Dimension

A point in time can be used to further describe the status of an element, e.g., a real-world object, a recommender item or the item's metadata. In other words, an element has a specific state at a particular point in time. This state might change and re-occur. The time dimension is appropriate to describe this process of change and thus determines each element's state at a particular time. The timeliness of social data, for instance, "is very limited as the users' interests tend to change quickly over time. Time is considered as an important factor when building social *Recommender Systems*" [58].

Campos et al. [57, p. 73] describe the inclusion of time attributes in *Context-Aware Recommender System*, such as "time of the day, day of the week, and season of the year", as an elementary

aspect of *Time-Aware Recommender Systems (TARSs)* . Its mathematical representation is:

$$F : U \times I \times T \rightarrow R, \tag{3.5}$$

where the concrete relevance score $r \in R$ depends on the user–item–time triplet $< u, i, t >$ containing user data $u \in U$, item data $i \in I$ and time information $t \in T$. Figure 3.2 shows an example of a *TARS* model that determines a relevance score for the user with $id = 101$, item $id = 7$ and time label with $id = 1$ that is $R(101, 7, 1) = 6$ (this example is borrowed from Adomavicius and Tuzhilin [10, p. 227]).

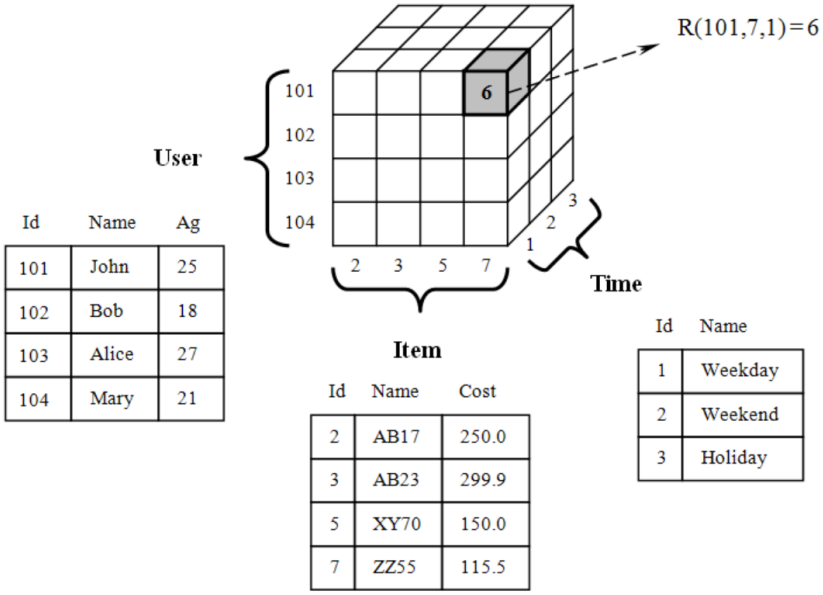

Figure 3.2.: Example of a *TARS* utilizing a three-dimensional model for $User \times Item \times Time$. This example is taken from Adomavicius and Tuzhilin [10, p. 227] (the user Ids are corrected due to an error in the original figure).

In 2003, Tang et al. noted as one of the first researchers the importance of considering temporal effects in the items' metadata for improving *Recommender Systems* [260]. They noted the effect of different movie production years on the users' rating habits.

Time Attributes

Time events can be characterized as ordered numerical attributes. Thereby, they depend on the application area and unrestricted numerical, numerical in intervals or the new devised numerical in repeating interval types are used (see Section 2.2.1.1). The last attribute type is applicable for focusing on specific time units, such as hours, days, months or years, as it forms a hierarchy of time units, such as 60 minutes represent one hour, 24 hours represent a day, and so on. "This hierarchical structure and the fact that time is a continuum, lead to a cyclic conception of time where its values repeat periodically" [57, p. 73]. Depending on the application area, it might be useful to work with the entire available numerical range (e.g., by using timestamps) or to represent

a point in time or a period by a category label (e.g., "morning") to cluster similar points in time.

Time Labels

In 2009, Baltrunas et al. introduced a new approach to considering contextual labels, including for time periods. Thereby, these authors split their items into subsets representing significant differences in the ratings. "For each of these items, it splits the ratings into two subsets, creating two new artificial items with ratings assigned to these two subsets" [26]. Later in the same year, they extended their research and split the users' feedback data into "overlapping sub-profiles" for different context and time labels [25]. This approach increased the accuracy of the implemented *Collaborative Filtering* method after identifying the optimal split time and thus shows the improvement to recommenders given by considering time.

Unfortunately, the approach focuses only on categorical time labels – such as "morning", "Wednesday" or "summer" – and does not consider any dynamically and continuously changing time intervals or even the time in between two labels. Since a category label reduces the information space that might be useful for further calculations, it would be more flexible to persist with the exact timestamp of an action instead of a reduced label. The usage of disjunct time labels, in turn, makes it easier to incorporate other contextual dimensions as well [57, p. 78].

Following similar considerations, Campos et al. classified *Time-Aware Recommender Systems* according to "categorical time-aware approaches", "continuous time-aware approaches" or "time adaptive approaches" [57, p. 76]. The latter approach takes time information into account to adjust certain algorithm parameters dynamically in a "more subtle way". In contrast to the other two approaches, time adaptive systems do not directly influence a rating prediction for the target recommendation time [57].

Time Dependency

In contrast to Campos' term of *Time-Aware Recommender Systems*, Lathia et al. formalize "*CF* as a time-dependent, iterative prediction problem. [...] A further evaluation of adaptive-CF would therefore also encompass the variation in recommendations that results from user parameters being constantly updated" [172]. Time dependency refers to mathematical functions of time, where it is possible to determine for each point on the y-axis a point in time on the x-axis. Subsequently, a *Recommender System* that allows allocating of a point in time to a timely appropriate relevance score will be called a Time-Dependent Recommender System.

3.5.1. The Effect of Time on the User Data

As time played a minor role in *Technology Enhanced Learning* so far, the related work of *Time-Aware Recommender Systems* mostly focuses on more general domains. However, some approaches have been published and the general concepts and findings are also applicable to educational *Recommender Systems*.

He et al. [123] analyze users' preferences and different resulting neighborhood groups over different time intervals. They underline the importance of context information for forecasting

user preferences because interests are different when various time intervals are considered and are "changing along with time rolling around" [123]. They note that existing algorithms focus on the users' similarity by considering all history interests. However, a *Recommender System* should particularly consider the periods during which users have overlapping interests.

Academic Datasets

Since time seems to play a crucial role in forecasting activities, why was the time aspect not considered previously? He et al. noticed that in most cases the "rating's timestamp is user's history impression, not the time when user saw it" [123]. Thus, the rating time is not necessarily the same as the consumption time. For movie *Recommender Systems*, for instance, users provided ratings even years after watching that movie. This circumstance makes it hard to analyze the time effect in the already obtained datasets as most academic datasets are based on ratings without any information on the difference between consumption and feedback time. Neither the popular Netflix Prize [34] nor the academic Movielens dataset [120] contain consumption time information. Nevertheless, even with these datasets, it is possible to prove the influence of time on ratings.

Types of Temporal Effects

Koren et al. [159] identified temporal effects in the Netflix Price dataset. One effect emerged in early 2004 (over four years since the first Netflix rating has been provided) when the average number of movie ratings increased suddenly. A second temporal phenomenon in the Netflix data is the correlation between age and the rating of a movie. The older a movie (measured time span since the first rating), the higher the average rating. Additionally, Xiang et al. identified four time factors that influence the recommendation of movies [286]:

1. **Time bias shifting:** The interest of a whole society changes with time. This effect is particularly present over the limited time period of a course. For instance, at the end of the course, all learners simultaneously learn the same topics.
2. **User bias shifting:** Users may change their feedback habits over time. Also this effect is observable for individual learners.
3. **Item bias shifting:** The popularity of items changes over time which can be observed during lectures.
4. **User feedback shifting:** Users may change their attitude towards some types of items.

Subsequently, the change of Netflix' average ratings can be classified as an effect of time bias shifting and similar effects might also be observable in *Technology Enhanced Learning*.

3.5.2. Existing RS Approaches concerning the Time Dimension

Xiang et al. [287] differentiate between short-term and long-term user preferences in a session-based temporal graph. Long-term preferences are based, as usual, on all past user ratings. Short-term preferences, in contrast, consist only of the ratings in the user's current consumption session. They proved the effectiveness of the method using two real datasets regarding citations and social bookmarking and improved the previous state-of-the-art recommender by 15% - 34%.

"[...] most research on time-dependent recommendation engines have been done in the area of movie predictions: For instance, a time-context based Collaborative Filtering algorithm, [123] proposed by Liang He, describes the inclusion of rating time in the computation of predictions for movie ratings. And Zhang et al. [300] describe an approach to consider changes in users' interests when recommending the items for the users. While a novel K-nearest neighbor algorithm [183] finds time-based neighborhoods of a user"

Cf. [161, p. 502][45].

The MovieLens dataset is used for evaluation of He's approach and the error and recall measures prove the good quality of the approach. On each dataset, the proposed *TARS* algorithm outperforms the conventional user-based filtering algorithm [123].

Zhang and Liu [300] used a *Item-based Collaborative Filtering* algorithm "based on Time Period Partition" to consider changes in users' interests in recommending the items for the users. They identified that most significant patterns only occur over specific time intervals. Therefore, the user's rating history was divided into several periods. Each period was treated as a simple sub-community and recommendations were only calculated based on the data in the given sub-community. The evaluation of the MovieLens dataset shows that the proposed algorithm outperforms even another Time–weight *Collaborative Filtering* algorithm (which was presented in [89]) [300].

3.5.2.1. Time Decay for Past Ratings

The time decay effect is a special form of user bias shifting and is part of Ding and Li's research on movie recommendations: Recently rated items "have a bigger impact on the user's prediction than an item that was rated [a] long time ago" [89, p. 491]. Thereby, they introduce a time function $f(t)$ for the weighting process representing "a monotonic decreasing function, which reduces uniformly with time t and the value of the time weight lies in the range (0,1)" [89, p. 487] as shown below:

$$f(t) = e^{-\lambda * t}. \tag{3.6}$$

e-Function

This e-function takes the decay rate $\lambda = \frac{1}{T_0}$ and the current time t into account. T_0 is the half-life parameter that reduces the total weight of $f(t)$ by $\frac{1}{2}$ in T_0 days. At the same time, Ding and Li's algorithm learns "users' rating behavior to find the appropriate personalized parameter for each item cluster" [89, p. 491]. Different error measurements show an accuracy improvement in all analyzed settings. As this approach sounds reasonable for incorporating a memory effect, it is considered for the *Smart Learning Recommender*.

Divergence in the Literature

Campos et al. [57] found divergence in the literature when considering the effect of time

[45] This excerpt is part of the related work section on a time-dependent Learning Recommender System and was exclusively written by Christopher Krauss.

decay: While Ding et al. [89] received better recommendations when only considering the latest ratings of a user, the data processing of the Netflix Prize [159] showed better results without time decay. Campos shows "that the usage of information near to the recommendation date alone can help improving recommendation results, with the additional benefit of reducing the information overload of the recommender engine" [56].

A similar approach concerning time decay is the "Time-Based K-Nearest Neighbor Collaborative Filtering" [183]. Liu et al. call their decay parameter the "Importance Degree of Items" which improved the precision by up to 13%. Lu et al. proposed a spatiotemporal model enabling "an adaptive estimation procedure that emphasizes recent user ratings more than his/her past behavior" [185, p. 13]. Moreover, they incorporate also extra information, such as demographic side information and implicit feedback data, as additional user and item factors. These factors are modeled in a time-dependent manner to represent changing attributes (e.g., preferences) over time. They applied Matrix Factorization as well as Kalman Filtering [121] and evaluated the system, among others, on the MovieLens dataset. Thereby, the incorporation of the spatiotemporal model reduced the prediction errors by 1.4% compared to a traditional model. The *Smart Learning Recommender* introduces a similar spatiotemporal data representation, but instead of an exclusive *Collaborative Filtering* approach, it is a *Hybrid Filtering* approach with a major focus on item attributes (for *Content-based Filtering* and *Knowledge-based Filtering*) rather than other learners.

3.5.2.2. Forgetting in Technology Enhanced Learning

The first models that aimed at abstracting and forecasting knowledge decays were developed in the late 19th century. Thereby, Ebbinghaus developed a first, but in parts still reliable, model of human forgetting [98]. It is based on observations in experiments regarding random syllables form fantasy words which have been learned (and forgotten) by humans over three years. In regular intervals, the gained and still available knowledge is tested which results in an average progress of forgetting over time. An extension of this approach (that also incorporates some digital media parameters) is utilized as a context factor for the *Smart Learning Recommender* in this dissertation. Moreover, the idea of automated forgetting of information for improving Data Mining is also incorporated in different variations within this work.

Some research highlights positive aspects of forgetting, especially forgetting in the area of big data. For instance, "forgetting" or "trashing" might be a necessary instrument when storing or processing information in huge datasets in order to handle less data in total and to improve the overall performance [104].

Regarding educational *Recommender Systems*, "there has been research on forgetting for scheduling practice [140] [, but] no systems [...] have included forgetting into open learner models, as done in PAL3" [254, p. 492]. PAL3 is a learning platform that presents a digital avatar to the learner

that gives hints for life-long learning scenarios – even for *Learning Objects* that might have been forgotten. Thereby Swartout et al. adapted the forgetting model of Averell [23], an exponential decay model with a non-zero asymptote which represents the fate of memories. The model of Swartout et al. "attempts to estimate both the asymptote and the current mastery simultaneously, where each observation is weighted based on the expected amount of forgetting (i.e., three high scores each a month apart raise the asymptote greatly, but three high scores a minute apart will raise current mastery but do little to change the asymptote). Forgetting is applied every time a new score is added or when calculating mastery after the learner has not practiced [...] at least one day" [254, p. 494].

3.5.2.3. Time Weights

Adibi and Ladani define the similarity between users as "group similarity" [7]. Thereby, a group of items is represented by common attributes, such as the same genre. This means that users are compared not by their rating of specific items, but their average rating of genres. They analyzed the effect of weights for rating timestamps in similarity groups. The higher the timestamp, the higher the weight $w(r)$ for rating r:

$$w(r) = (t_r - t_{max} + 1)^b, \tag{3.7}$$

where t_r is the current rating's timestamp and t_{max} is the timestamp of the oldest rating of the same user. "b adjusts intense of forgetting, the way whatever its value to be higher, recent ratings will have more influence in calculations" [7, p. 254-255]. Evaluations with the MovieLens dataset show a significant improvement of the error measures and the coverage values, especially for cold start users. This approach is borrowed and adapted for the time weighted concept of the *Smart Learning Recommender*.

Chen Dongtao [90] used user profile weights as well as time weights to improve the neighborhood-based recommendations of MovieLens items. In a first step, he takes similar profile information (such as users' age, gender and occupation) into account. In a second step, Dongtao gives recent ratings a higher weight with the help of a "monotonic increasing function [with] forgetting ability". The older the ratings, the lower the weight. While each step (incorporation of user profile weights as well as time weights) outperforms the traditional *CF* for its own, the combination of both is even better.

Time-weighted Slope One

As stated in Section 2.2.2.1, the Slope One algorithm is a frequently used *Item-based Collaborative Filtering* approach to predicting the missing user ratings. Jiang and Lu extended the idea of a Weighted Slope One algorithm introduced by Li et al. [177] and increased "the weight of the user's recent behavior and give priority recommended for those items which [are] similar to user's recently favorite items" [142, p. 2296]. The resulting improved Time-weighted Slope One Algorithm takes

user bias shifting into account and increases the accuracy compared to the original version by 3.1%. A novel comparison of the traditional and the time-weighted Slope One, both applied in the same *TEL* setting, is presented in more detail in Section 7.2.1.

3.5.2.4. Time-based Recommender Systems in TEL

Verbert et al. analyzed different *Recommender Systems* that have been applied in the *TEL* domain. They identified also a number of *RSs* which incorporate time information in the calculation process [270, p. 325]. Lehsten et al. [173], for instance, store situation-related data (e.g., information on time and location) for the access of resources within their *LMS* to further understand these context dimensions (and without utilizing recommender techniques). Other researchers focused on the filtering of learning resources according to the available study time [73, 36, 293, 240]. However, in the strict sense of its definition, they cannot be seen as *TARS* algorithms.

Hermann [126] published an approach for recommending educational material by incorporating time features in a "contextual modeling" manner. Verbert et al. noted that Hermann was the sole author so far who developed a *Time-Aware Recommender System* for *Technology Enhanced Learning* [270]. This algorithm is called *Time-based Recommender Approach for Lecture Materials (TBRA)*. Hermann determined correlations between the download time of videos, animations (in Flash) as well as slides and particular dates, such as tutorial deadlines and exams. As the contents can be accessed independently from specific courses, he collected almost 5 million activity statements, called "TimePreferences" [126, p. 4] which consist of a user-item-time triplet for each download.

The recommender approach is based on a similarity measurement that is exclusively based on the point in time of item consumption. The Boolean function $p(u, i)$ becomes true when user u downloaded item i. The algorithm calculates the similarity of the two items $S_{i,j}$ for all users u who downloaded the two items i and j [126, p. 5]:

$$\forall u \in \{u \mid p(u,i) \wedge p(u,j)\}. \tag{3.8}$$

$$S_{i,j} = \frac{1}{\Delta t_{i,j} + \Delta d_{i,j}}. \tag{3.9}$$

Thereby, $\Delta t_{i,j}$ corresponds to the deviation between the download time t_i of item i and the download time t_j of item j that can be formulated as $\mid t_i - t_j \mid$. $\Delta d_{i,j}$ represents the time deviation of the first of the two downloads $min(t_i, t_j)$ and the current point in time $t_{current}$ that is defined as $\mid min(t_i, t_j) - t_{current} \mid$ [126, p. 4].

The result is a ranked list of similar items. Thus, this *Time-Aware Recommender System* always requires a reference item which is consumed by the user initially. In addition to the actual requested item, the learner gains hints for other materials that have been downloaded recently and often in

conjunction with an actual requested one. An evaluation shows a slight improvement of precision (87%) and recall (17%) compared to a classical *Collaborative Filtering* algorithm (precision = 84%; recall = 16%) [126, p. 5]. More details on this approach as well as some realization details for comparison reasons can be found in Section 7.3.1. However, this approach must be adjusted to fit the needs of a *TEL RS* without a reference item.

3.5.3. Drawbacks of TARS

A negative example for the incorporation of additional contextual factors in general *RS* is given by Van Setten et al. [268]. However, the effects should be similar in specialized *TEL Recommender Systems*. They identified a negative perception of the usefulness of their touristic points of interest recommendations when taking the features of last-time visits into account [268, p. 9]. While most research on time improved the recommender's performance significantly, a smaller increase in the accuracy or acceptance can also be a disadvantage: Xiang et al. use complex matrix factorization in *Collaborative Filtering* for low-rank ratings. They used the Netflix and the MovieLens dataset for testing and evaluation and improved the overall error measurements by a comparably low value of up to 0.00558 [286]. Given the computational costs incurred, a service provider must balance benefits and limits of *Time-Aware Recommender Systems* very well. In the worst case, a *RS* without the awareness of time can perform better in total and, thus, should be preferred.

An up-to-date recommendation might have another negative impact when it is recommended in an inappropriate situation. Betzalel et al. [75] note that the correct timing of proactive recommendations helps to improve the user acceptance, as – in some situations, e.g., while driving, chatting or watching a movie – proactive recommendations are inappropriate. Therefore, they developed and trained a classifier that predicts appropriate recommendation phases. Their *Recommender System* for points of interests shows the best results when combining a personalized and a non-personalized approach in a hybrid way. This dissertation will introduce a new measurement value for determining the timeliness of recommendations.

3.6. Learning Path/Sequence Recommender Systems

When users want to obtain a complete overview of all offered *Learning Objects* or to see the recommended order of the next items, traditional *CF* algorithms need to be extended. The identification of *LO* sequences is typically handled by a special group of *Recommender Systems*, as it requires knowledge about the relations between items instead of just predicting a relevance score per item. Shen and Shen [244] introduced a prediction model with a sequencing rule algorithm by taking a topic ontology into account. When the system identifies a lack of knowledge, appropriate contents are recommended. In this work, a novel Learning Path algorithm builds on

this foundation.

Item Sequencing

A common technique for the presentation of item relations is to create a database with directed graphs, including nodes for items and edges for their relations [21, 271, 291]. Thereby, different approaches are frequently utilized to create the needed item sequences. Six key approaches are described in the following list:

1. **Teacher's Sequence:** Teachers and educational staff have the best knowledge of the taught topics and the intended knowledge transfer. Thus, they model item sequences manually and in a pedagogical way [94]. Their model results in a high quality for the given learning setting because the *Learning Objects* are then curated by experts. However, this approach is time consuming for the content creators, because of the manual content management [21]. This approach on its own does not allow for the provision of personalized learning paths as the predefined didactic structure remains static.
2. **Content-based Analysis** allows for the generation of a topical structure without the supervision of humans. It rather analyzes the content, especially textual input, to automatically generate dependencies. Chen, for instance, uses semantic text analysis approaches (namely TF-IDF) to identify the most important terms within *Learning Objects* and, consequently, to cluster the items according to their topics [61].
3. **Constraint-based Approaches** are similar to manually defined structures in allowing for more choice in the personal learning directions with some predefined restrictions [21, 271]. For instance, the learners might appreciate the opportunity to freely choose the next learning item as long as all prerequisites are fulfilled.
4. **The knowledge-based Approach** is based on the previously acquired knowledge of the learner, e.g., by presenting a survey at the beginning of a course or by letting the user answer some related questions when accessing the content. According to the determined knowledge, well-known items are filtered out, and items with new topics gain a higher relevance [61, 291]. An alternative is to present learning objectives, as done by the *Learning Unit* in this work, according to the taxonomy of Bloom [40], and to allow the learner to provide self-assessments based on these objectives [305, p. 230]. The knowledge gained, in turn, can be represented as an ontology [62].
5. **Activity-based Analysis:** The set of user activities can be utilized to generate *Learning Object* dependencies based on the past consumption activities of learners [94]. Some researchers used the approach of ant colony optimization that corresponds to the traveling salesman problem [291, 189, 91]. The transition from one learning item to another yields a lower penalty, the more frequently the items are consumed in a particular order. This approach requires a sufficient number of users and interactions. Huang et al. [132] used a Markov Chain Model to identify automatically sequences of learning objects in past courses.

Moreover, the context of the collected interactions is important, as typical learning behavior is not necessarily generalizable [17]. Thus, the interactions should be used in the same context of learning path recommendations.

6. **Hybrid Combination:** Similar to *Hybrid Filtering*, the hybrid combination of approaches represents the most powerful class, as it overcomes the weaknesses of single approaches. For instance, the time-intensive teacher approach can be assisted by content-based analysis and interaction-based approaches can be combined with the knowledge-based approach to overcome the cold start problem [291]. The most common method is a combination of teacher-based or knowledge-based with a constraint-based approach, providing a curated but also – to some extent – personalized learning experience [21, 271, 61].

Published Learning Paths

Most publications allocate an important role to user feedback for learning path generations in terms of the learners' interests and knowledge levels. Drachsler et al. [93], who mainly focus on *CF*, note that explicit user feedback, such as ratings or tags, helps to identify paths in learning networks in a more efficient way than other input types. Voss et al. developed an adaptive sequencer that uses Matrix Factorization as the performance predictor [273]. With the same approach, Schatten and Schmidt-Thieme tried to "keep the contents in the Vygotskis Zone of Proximal Development (ZPD) [277], i.e., the area where the contents neither bore or overwhelm the learner" [237]. In each calculation step, the system selects contents with a predicted performance score that is most similar to the user's modeled score. Researchers of the Worcester Polytechnic Institute [288] enhance long-term retention of acquired knowledge by creating a Personalized Adaptive Scheduling System for retention tests.

Another – very recent – example of learning paths was realized by Nabizadeh et al. [199] which is introduced in detail in this paragraph: They restricted the item sequence to those *Learning Objects* that allow "obtaining the maximum possible learning score in a limited time" [199, p. 153]. Thereby, they focus on three sub-problems: (1) the generation of all possible paths based on the user's previous knowledge and a restriction of the total learning time, (2) estimates of the personal time for the generated paths and (3) scoring predictions for the paths. Thus, the generation is a knowledge-based approach [199, p. 155]. The previous learning speed of a particular learner is compared with the median learning time of others and the resulting learning speed is transferred to other items. The scoring prediction, in turn, is based on the average learner's assessment scores (0: fail; 1: pass) per *Learning Object*. Taking this information into account, the path generator recommends items with the highest average assessment score and the lowest estimated learning time out of all possible paths. The evaluation of this learning path approach shows low error values, but only compares the prediction accuracy of the time and scoring predictions. The accuracy values of entire recommended paths are not determined.

Missing Branches

While path creation algorithms are frequently utilized for the prediction of learning sequences

[21, 271, 61, 132, 291], none of them consider the presentation of alternative routes with path branches. However, the idea of a *Recommender System* is to offer different choices in a particular situation. This work will focuses on an approach that incorporates paths with branch alternatives which are recommended to the user.

3.7. Conclusions for Recommender Systems in TEL

This chapter introduced various approaches to educational *Recommender Systems* and discussed the main differences to traditional ones known from the entertainment sector and e-commerce domain. This leads to the definition of a specialized paradigm which should be followed when recommending learning items in closed courses. Afterward, four particular classes of *Recommender Systems* have been analyzed in detail: *Collaborative Filtering* for *Technology Enhanced Learning*, *Context-Aware Recommender System*, *Time-Aware Recommender System* and *Recommender Systems* for the prediction of learning paths.

Scientific Hypothesis /SH2.0/:

An educational *Recommender System* that recommends course items should respect the special paradigm for the recommendation of course materials. The paradigm comprises various aspects, for instance, a learning-oriented relevance score that is called the learning need, the necessity of incorporating multi-dimensional context attributes and the time dependency of the data that impacts the precision of the recommendations. If a *Recommender System*, that does not respect, or only partially respects, the special paradigm, is applied for the recommendation of course items, it would generate less precise Top-N recommendations.

Next Chapter

The next chapter introduces an infrastructure to deploy and evaluate an educational *Recommender System* in *Technology Enhanced Learning*. Open standards and specifications are applied in order to design a reference architecture for adaptive learning environments.

4. Reference Architecture for Interoperable Adaptive TEL Environments

About this Chapter

Most of the studies included in this dissertation have been performed with the help of a reference architecture that was created and adapted for use in a number of different learning settings[46]. This chapter focuses on the technical architecture which comprises components of three main classes: (1) User Interfaces, (2) Middleware and the Service-Logic Layer and (3) Databases and Repositories. Figure 4.1 gives a simplified overview of the reference architecture – in fact, the real applied architectures depend on the actual learning setting and the connected components (as described in this chapter). However, the figure is adequate to list and summarize the components and their general inter-relationships.

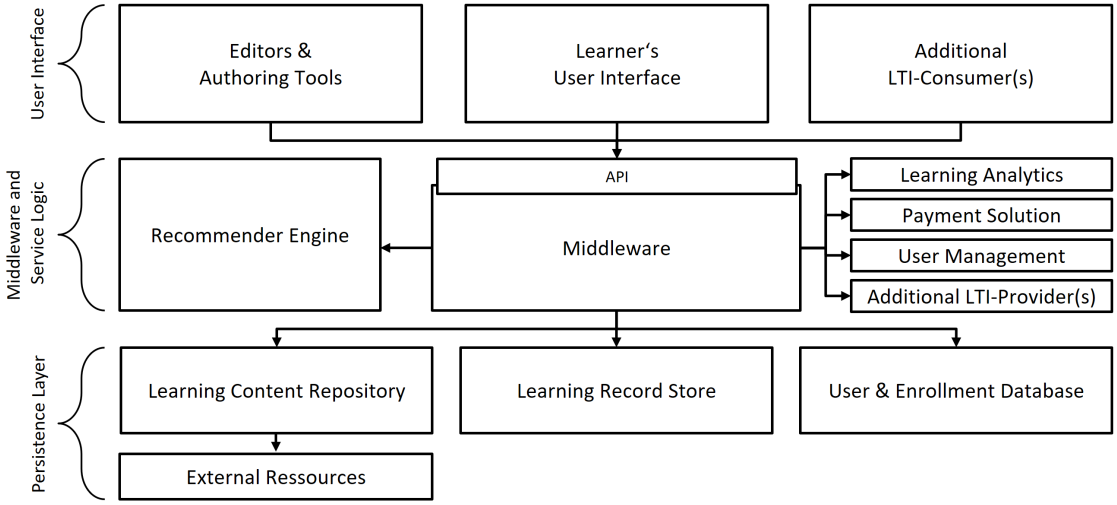

Figure 4.1.: Reference architecture for an interoperable adaptive *TEL* environment; Arrows indicate the main dependencies between the components. For example, the Learner's User Interface is connected to the Middleware which, in turn, requests data from the *Recommender Engine*.

All components are encapsulated by the Middleware. User interfaces, such as Editors & Authoring Tools, the main learning interface as well as additional components which render learning contents,

[46] An overview of the projects and other activities which led to this reference architecture is presented in Appendix B.1.

are connected to only one component: The Middleware, in turn, is connected to the service components, such as User Management, Payment Solution, Learning Analytics module and/or the *Recommender Engine*. Additionally, the data is stored in three different databases: a Learning Content Repository, which holds the course media as well as its metadata, a *Learning Record Store* for persisting activity data and a User & Enrollment Database for managing course participants. Learning media can also link to external sources, but the main metadata should be hosted internally for reasons of consistency. Depending on the implementation, each of the services may be connected directly to one or more databases. The main components of the reference architecture, their *Application Programming Interfaces (APIs)* as well as their data formats are introduced in the following sections.

4.1. Systems for Accessing and Managing Learning Contents

LMS and LCMS

Learning Management Systems (LMSs) are the central entry points to the educational content for learners and teachers. An *LMS* represents "the infrastructure that delivers and manages instructional content, identifies and assesses individual and organizational learning or training goals, tracks the progress towards meeting those goals, and collects and presents data for supervising the learning process of an organization as a whole" [257] (cited in [280]). Moreover, a *Learning Management System* "delivers content but also handles course registration and administration, skills gap analysis, tracking and reporting" [112] (cited in [280]). Thus, it is a digital presentation of traditional classroom environments. A *Learning Content Management System (LCMS)*, in turn, is an extension of an *LMS*. In allusion to a *Content Management System (CMS)*, where the creation and management of content are key aspects, an *LCMS* has an additional focus on the creation of learning content and its management.

In Europe, Moodle[47] is the *LMS* market leader with about 65% market share in higher education[48], while in the United States and Canada the market is much more fragmented and Blackboard Learn[49] is the leader[50]. An analysis of the market shares in these regions is presented in Appendix B.19.

Open vs. Proprietary

Especially for further processing of metadata in learning environments, it is critical to build on existing standards and specifications in order to provide the best-possible cross-platform support for the required components [202]. An analysis conducted by LISTedTECH[51] indicates that there is an almost equal distribution between open and proprietary learning technologies in higher education

[47] Moodle. See: `https://moodle.de/` (Accessed: 20.02.2017).
[48] European LMS Market. See: `http://listedtech.com/european-lms-market/` (Accessed: 21.11.2017).
[49] Blackboard Learn. See: `http://www.blackboard.com/` (Accessed: 20.02.2017).
[50] See: `http://mfeldstein.com/state-higher-ed-lms-market-spring-2016/` (Accessed: 21.11.2017).
[51] Overview of "Free LMS or Open Source LMS used in Higher Ed" by LISTedTECH. See `http://listedtech.com/free-lms-or-open-source-lms-used-in-higher-ed` (Accessed: 20.11.2017).

(51.4% open source and 48.6% proprietary). While some platform providers, such as edX[52], try to feature their proprietary components and formats by offering them exclusively, an institution that uses this platform is thus tied to the offering of a particular provider. Transfer of content or data to another platform often becomes problematic because of lack of interoperability. This is why a large community of educational institutions supports open specifications. Additionally, Drachsler et al. [92] raised the issue of missing comparability between educational datasets – which is a particularly significant issue for *Recommender Systems* and is why the next section focuses on open specifications in *Technology Enhanced Learning*, which are the basis for *Recommender Systems*.

4.2. Specifications for the Description of Users, Contents and Activities

SCORM

The *Advanced Distributed Learning Consortium (ADL)* published the *Sharable Content Object Reference Model (SCORM)* in early 2000. *SCORM* was the first generally accepted, open specification for a unique handling of learning contents and it came with a huge set of definitions [41]. The three core books reference to:

- *SCORM Content Aggregation Model (CAM)* for managing learning contents, handling its structure, searching and discovery as well as the definition of content packaging and transfer.
- *SCORM Sequencing and Navigation (SN)* for handling content sequences, learning activities and navigation data.
- *SCORM Run-Time Environment (RTE)* for the definition of requirements for *LMS* to display *SCORM* contents.

With *SCORM*, learning providers manage their learning contents, render them into supported *LMSs* and represent user activities consequently in a standardized way for the first time. However, the need for new features, such as adaptability and learning analytics in learning platforms, gradually indicated the inflexibility of these specifications. Even its final, 4th, edition of 2004 could not cover all upcoming requirements[53]. *Learning Objects*, for instance, cannot be accessed independently from the given didactic structure. This, however, is a key requirement for *Recommender Systems*. Moreover, reporting features, such as user tracking, are given less consideration in the 4th edition of *SCORM* because *ADL* was working on another specification that mainly focuses on the collection of learning activity data.

xAPI

The new *ADL* specification is called *Experience API (xAPI)* (formally known as Tin Can). *xAPI* was released in 2013 and describes the recording and transfer of all types of learner activities

[52] edX. See `https://www.edx.org` (Accessed: 20.11.2017).

[53] See: `https://www.efrontlearning.com/blog/2013/04/why-scorm-2004-failed-what-that-means-for-tin-can.html` (Accessed: 07.12.2017).

within educational software [149]. Thereby, each *xAPI* statement represents a micro-ontology that consists of a learner, a verb and a target object regarding the learning content. Moreover, additional metadata can be added as contextual data:

> "A typical statement consists of the three properties: 'Actor', 'Verb' and 'Object'. An xAPI statement can also carry the optional properties 'Context' and 'Results' containing more information for new insights like in the following statements:
>
> - 'StudentA (Actor) completed (Verb) Question1 (Object) in the context of Quiz1 in Course1 and the result is success with 2 attempts based on the raw score of 80 with a max score of 100 and a scale of 0.8'
> - 'StudentB stopped VideoY started at position 00:01:30 in the context of LearningUnit2 of Course1 resulting in duration of 00:01:42"'
>
> Cf. [166, p. 476][54].

The *xAPI* activities are the key elements for further adaptability features, as they allow for the development of complex user profiles with an increasing number of statements[55]. In parallel, the *Instructional Management System Global Learning Consortium (IMS)* published a specification that also introduces an ontology (Actor / Action / Activity) for storing user activities – called Caliper[56]. An initial comparison of *xAPI* and Caliper indicates that there are a lot of similarities and recommends to both *ADL* and *IMS* to better highlight their complementary features: "xAPI and Caliper are NOT equivalent. Therefore, the adaption of the current versions should not be 'one-or-the-other', instead it is a 'horses-for-courses' decision" [57]. The here introduced reference architecture builds on *xAPI* for tracking the learners' activities.

Learning Record Store

These activity data need to be stored in a particular database – called the *Learning Record Store (LRS)*. An *LRS* collects all of the statements of users regarding learning contents without recording the user or item metadata but only references to them in other databases. A reference is presented as *Uniform Resource Identifier (URI)*. A *URI* links to personal data of the user (e.g., username, affiliation, etc.) in a user management system or to learning content data in a Learning Content Repository. One example of an open source *Learning Record Store* is Learning Locker[58]

[54] The paper extraction was primarily written by Truong-Sing An who is one of the project partners and describes *xAPI* statements in the Smart Learning Project. The collected *xAPI* statements are processed by a Learning Recommender to predict appropriate learning objects, which is described by Christopher Krauss in the same paper.

[55] An example of an *xAPI* statement collected via the Smart Learning Infrastructure is presented in Appendix B.24.

[56] Just before the release of *xAPI* and Caliper, Niemann et al. [203] listed four other usage data formats where especially the schemes "Activity Streams" and "Learning Registry Paradata" show a lot of similarities to *xAPI* and Caliper regarding the data structure. While the aim of these schemes is similar, they seem to be less popular than the specifications of *ADL* and *IMS*.

[57] xAPI/Caliper Comparison. See: `https://www.imsglobal.org/initial-xapicaliper-comparison` (Accessed: 09.05.2017).

[58] Learning Locker. See: `https://www.learninglocker.net` (Accessed: 10.03.2017).

which is based on the document-store MangoDB[59]. For instance in the Smart Learning project, the *xAPI* activity data are stored as *JavaScript Object Notation (JSON)* in the Learning Locker.

Content Access

The previously mentioned *Instructional Management System Global Learning Consortium* published the *Learning Tools Interoperability Specification (LTI)* to provide mechanisms for an interoperable access to learning materials [5]. Thereby, each *Learning Object* can be rendered by an *LTI* consumer which requests the contents via a standardized *Representational State Transfer (REST) API*. The parameters for content selection (e.g., the content identifier as *URI*), content presentation, user access rights and content protection are transferred to a server-side component – the *LTI* provider. The *LTI* provider, in turn, delivers *Extensible Markup Language (XML)*-compliant data for the presentation of the learning material on the consumer side. As *HTML* is a subset of *XML*, *LTI* can be used to provide *Learning Objects* as *World Wide Web Consortium (W3C)* compliant web contents. The idea is to abstract the contents' persistence layer and metadata structure from its graphical presentation and thus provide a standardized interface between the two. *LTI* allows for the presentation of *Learning Objects* independently from any specific *LMS* or *LCMS* or even to display learning contents in the absence of a learning platform. This independence is crucial for *Recommender Systems*. The middleware which is essential for the reference architecture offers an *API* which acts as *LTI* provider.

Learning Content and Metadata

One of the main aims of *SCORM* was to provide a reliable format for the persistence of learning contents and related meta information. However, *SCORM* is highly complex with all of its sub-specifications and comes as a one-fits-all solution without the flexibility to replace specific sub-specifications. Thus, the *IMS* defined a set of specifications for interoperable and also flexible handling of learning data. The *IMS Common Cartridge Specification (CC)*, version 1.2 published in 2015, represents course material in standardized formats to be used in a wide variety of *LMS*. It comes in two different settings: as a full profile with the whole range of *IMS* sub-profiles and a thin profile only containing the previously introduced *LTI* specification, Web Links and *IMS Learning Resource Meta-data Specification (LOM)* [3]. For the reference architecture, the full *IMS CC* profile is of interest, as it includes all important definitions for a *TEL RS*: For instance, *LOM* defines formats and types (in *XML*) for the standardized description of learning media. The attributes for content prerequisites, learning goals as well as intended learning time are of particular interest for *TEL Recommender Systems*. Similarly to *LOM*, the *IMS Question and Test Interoperability Specification (QTI)* defines the structure of exercises, quizzes and tests. Finally, the *IMS Content Packaging Specification (CP)* "focuses on the packaging and transport of resources" which is critical for the exchange of contents between various platforms "but doesn't determine the nature of those resources [... to] aggregate content in an unlimited variety of formats" [2][60].

[59] MangoDB. See: `https://mangodb.com` (Accessed: 10.03.2017).

[60] An introduction of alternative specifications by other organizations is presented in Appendix B.20. However, those contributions are outside of the scope of this dissertation.

4.3. Example Architecture for the Smart Learning Environment

The technical components of the Smart Learning project build the basis for the collection of learners' activity data for further evaluations. At the same time, the predicted recommendations are presented via the user interfaces. This is why the core components of the environment that have an impact on the *Recommender Systems* are introduced in the following subsections. This comprises the *Learning Companion Application (LCA)* as a user interface and as an entry point for the learners and teachers[61], the content repository with media assets and descriptive data on *Learning Objects* as well as the *Learning Record Store* which holds the behavioral data. In addition, the *Recommender Engine* is described. However, the actual algorithms are the subject of special focus in this work and are introduced in subsequent chapters.

4.3.1. Description of the Learning Companion Application

The Smart Learning project was first introduced in 2015 [165, 305] and focuses on meeting the needs of the Chamber of Crafts in Berlin. The project introduces digital media for vocational training that will be presented to learners in a novel mobile application – the *Learning Companion Application (LCA)*. The *LCA* is a learner and teacher interface and offers some features of a *Learning Management System*[62].

> "One focus of the Smart Learning project is to provide a reusable generic infrastructure for various users with different client devices, for different courses covering several topics - not restricted to institutions like [the] Chamber of Crafts, but also usable by universities and adult education centers. While users are still managed in the *LMS* [Moodle], *Learning Objects* are stored only once and can be shared by and accessed from various *Learning Management Systems*.
>
> [...] The *Learning Companion Application* plays a key role. It is the entry point for students to access courses, *Learning Objects*, and lecture dates as well as to get recommendations for the next best contents to be learned and triggers the tracking of all relevant user interactions. It is a responsive web application to be displayed on regular modern desktop web environments, but especially on smartphones and tablets to enable mobile learning. The application gives everywhere-and-everytime-access to all *Learning Objects*. [...] Moreover, teachers use the *LCA* to get access to the Learning Analytics module.
>
> The *LMS* [Moodle] is used to register and manage all users and offers discussion forums. In order to allow a consistent interaction with all components, the students (and teachers) credentials of the existing *Learning Management Systems* are required to authenticate at

[61] Appendix B.21 contains a stakeholder analysis and use-case diagram for the environment.
[62] An alternative user interface, which is based on the defined reference architecture, is the FOKUS-Akademie which is introduced in Appendix B.22.

the *Learning Companion Application*. This kind of single-sign-on approach is implemented in the middleware and transparent to users"

Cf. [168, p. 14-15][63].

Technical Insights

From a technical point of view, the *Learning Companion Application* is a responsive web application based on *W3C* specifications: *HTML* 5, *CSS* 3 and *JS*. It is developed for the node.js runtime[64] enabling ECMAScript 6 (it uses the JSPM[65] package manager and a gulp[66] deployment workflow). A description of the user interfaces, in terms of the visual presentation of the different components, of the *Learning Companion Application* is presented in Appendix B.23.

4.3.2. Content Repository and Metadata

Item Data

A core aspect of the reference architecture is the utilization of open specifications which make its components exchangeable and re-usable. Thereby, the item assets and metadata are stored in well-specified, open formats in the learning content repository.

"The repository acts as a digital asset store, which essentially holds course structures, *Learning Objects* and their metadata. At the lowest level, a *Learning Object* is a simple document (technically in *HTML*), a video, a screencast, a test and so on, all with at least one learning objective. Low-level *LOs* are stored as *LTI*-Tools [5] as to integrate them with different *LMS*. Moreover, questions and tests are specified according to the *IMS QTI* specification [6]. Low-level *Learning Objects* can be bundled into bigger *Learning Objects*, and this iteration can be repeated. In the current energy consultant course, low-level *LOs* are combined in *Learning Units*, *Learning Units* in sections and a few sections make up the course. That way low-level *LOs* can be reused in several courses. A so-called manifest file is created to bundle the *LOs* together. A player that is presently stored in the repository renders the learning units and *QTI* specified tests. Further, the player generates automatically self-assessment questions using the learning objectives contained in a learning unit. The metadata associated with a low-level *LO* contains among others its learning objectives, at least one as mentioned above, average study time defined by instructors and prerequisite *LOs*. These data are stored using the *LOM* specification [1]. When *LOs* are combined, the metadata of the whole is generated automatically from the parts. A course structure is stored following the *IMS Common Cartridge Specification* [3]"

Cf. [168, p. 15] [67].

[63] This paper excerpt was written by Christopher Krauss, Prof. Dr. Agathe Merceron and Sinh Truong-An. Stefan Mueller of the Beuth University worked additionally at the underlying concept at an early project stage.
[64] Node.js. See: `https://nodejs.org/en/` (Accessed: 26.11.2017).
[65] JSPM. See: `https://jspm.io/`
[66] gulp. See: `https://gulpjs.com/`

Authoring Tools

The way of storing *Learning Objects* is not only crucial for the connected *LMSs* as it must only interpret *LTI* tools, but also for the *Recommender System* that needs to process item metadata. The algorithms, especially for *Content-based Filtering*, need to access descriptive data of the items to match their attributes. The data are initially created with the help of authoring tools.

> "Different editors have been implemented as easy to use web applications for instructional designers. A *LOM*-Editor allows specifying the metadata of any existing *LO* and to store the corresponding file in the repository. A *QTI*-editor allows creating questions, to bundle them into tests following the *QTI* specification and to store them in the repository too; presently seven types of questions are available: choice, choice multiple, extended text, text entry, numeric, matrix, and order. Finally, a *LO*-Editor allows bundling *LOs* into bigger ones and generating the metadata file automatically as written above"
>
> Cf. [168, p. 15] [68].

Learning Objects and *Learning Units* that are presented via a *LTI* consumer also contain client-side scripts[69] to report the user interactions to a predefined *Learning Record Store*. This way of integrating learning materials enables *LMS*-independent tracking.

4.3.3. Learning Record Store

For the persistence of activity data, such as the learner's feedback on items as known from *Recommender Systems*, the Smart Learning project makes use of established approaches in the *Technology Enhanced Learning* domain:

> "Users' interactions with any *LO* are stored according to the opt-in procedure chosen by the user. Interactions are persisted using the *xAPI* specification [149] in the free *Learning Record Store* called learning locker. The recommendation engine and the learning analytics service load the needed interaction data in regular intervals to determine students' performance. The [recommender] aims at identifying the next best, most suitable *Learning Object* for the requesting student based on the calculated knowledge level and learning need for that item. The learning analytics service, in contrast, is designed for other stakeholders. So, teachers can observe the overall progress and performance of students and figure out weaknesses in learning and understanding.
>
> [...] A key role in connecting the users' interaction in *LCA* with the learning analytics

[67] While Christopher Krauss contributed to this paper excerpt and the general idea, it was realized, first and foremost, by the team of Beuth University – especially Prof. Dr. Agathe Merceron, Sinh Truong-An and Francois Dubois. However, these concepts are important for the development of the *RS* as they describe the items' metadata.

[68] The editors and authoring tools have been implemented by the team of Beuth University. The text excerpt was also written by them.

[69] JavaScript was used to send *xAPI* statements via *Asynchronous JavaScript and XML (AJAX)* calls.

> service or recommendation engine is attributed to the formal and informal activity statements reflecting the collected user data. In recent years, the *Experience API* [149] with its *xAPI* statements continuously moves in the academic focus supporting long-term data mining"
>
> Cf. [168, p. 15][70].

As the middleware retrieves encapsulated *LTI* tools, a script for generating and sending *xAPI* statements is directly integrated into the *HTML* representation of the learning contents. This is why these tracking data are stored separately from the presenting *LMS*. However, the *LMS* (e.g., the *LCA*) can additionally send further *xAPI* statements, such as for logging in/out or clicking on recommendations. *xAPI* supports a list of 30 verbs in its *ADL* Vocabulary[71]. Currently, 10 of the available verbs are triggered in the Smart Learning environment which are described as follows:

1. **Initialized**: This statement is triggered when a *Learning Unit* or *Learning Object* was opened, or a video was started.
2. **Exited**: This statement is triggered when a *Learning Unit* or*Learning Object* was exited.
3. **Abandoned**: This statement is triggered when a user exited an *Learning Unit (LU)* or *LO* unexpectedly (e.g., by closing the browser).
4. **Answered**: A user answered a *QTI* question or provided a self-assessment. The context property contains a score value which describes the level of success on a normalized scale (in the range [0,1]).
5. **Downloaded**: A user downloaded a file, typically a PDF.
6. **Suspended**: This statement is triggered when a user paused a video before its end.
7. **Resumed**: This statement is triggered when a user resumed a previously paused video.
8. **Completed**: This statement is triggered when a user watched a video to its end.
9. **Waived**: This statement is triggered when a user sought material within a video (changed the position of the progress bar manually).
10. **Commented**: This statement is triggered when a user wrote a comment in the discussion forum.

An example of an *xAPI* statement collected in the Smart Learning Infrastructure can be found in Appendix B.24. Since these reports represent the learning behavior, they can be seen as user feedback (the user–item relation in the user–item matrix) of a *Recommender System*.

[70] At the time of developing this paper, first and foremost, Sinh Truong-An was responsible for the design of the persistence of behavioral data in the open source *Learning Record Store*. Different requirements of *LCA* led to an adaption with some specific verbs that are needed for the learning *RS*.

[71] *Advanced Distributed Learning Consortium* Vocabulary. See `http://xapi.vocab.pub/datasets/adl/` (Accessed: 06.07.2017).

4.4. Architecture of a TEL Recommender Engine

From a black box perspective, the *Recommender System* receives item metadata as well as user activity data as input and provides Top-N lists of item recommendations as output. Additionally, the relevance scores and the predicted knowledge levels, as well as algorithm explanations, are also provided as output.

Architecture

The arrows in Figure 4.2 visualize the dependency direction of the components. For instance, a *Learning Management System* (e.g., the *LCA*) calls the middleware which delegates the request to the *API* of the *Recommender Engine* in order to obtain a list of recommendations. The *RS* contains an *API* layer which delegates the request to the internal service layer. Depending on the particular algorithm, the service layer loads the required data via *Data Access Objects*, processes the data with one or more algorithms and returns the Top-N item list to the *API* layer that sends that list as a response to the front end that initially called the *RE*. In this work, different *RS* approaches in numerous settings and adjustments have been developed and analyzed. Details of the algorithms are presented in Chapter 7 and Chapter 8.

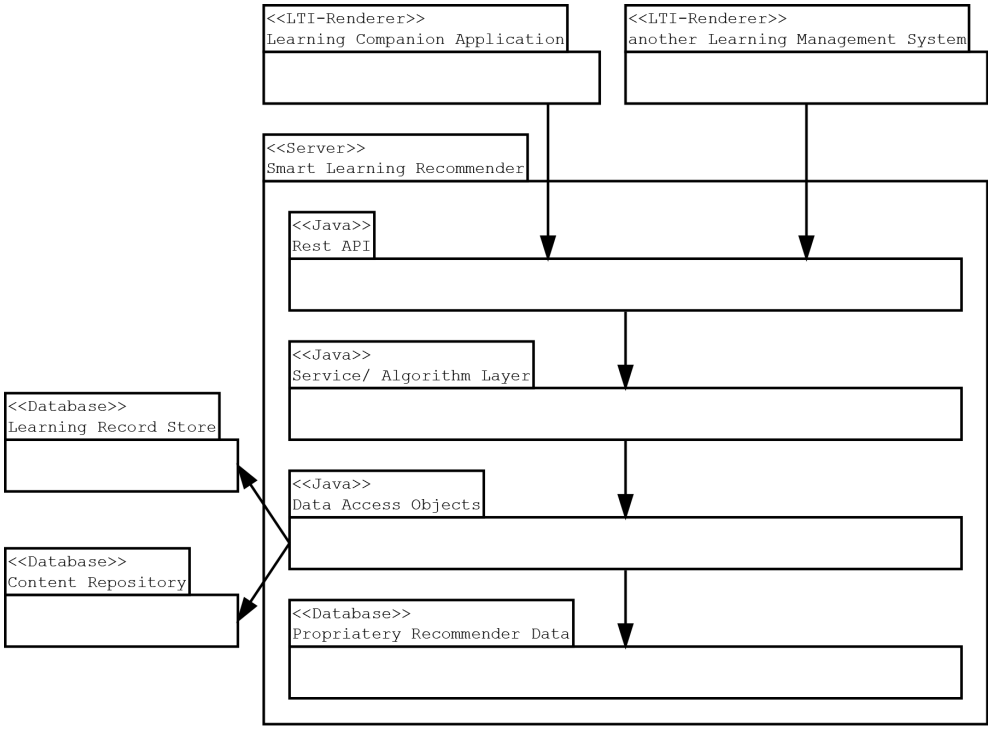

Figure 4.2.: Architecture of the *Smart Learning Recommender*; arrows indicate the dependencies of components.

SLR Data

While open specifications, such as *xAPI* or *LOM*, are required to exchange data between different components, the specified data presentation (in *XML* or *JSON*) is too exhaustive, complex and even contains irrelevant data for efficient data processing. The *Learning Record Store* consists of a

database where the data of every user–item transaction is stored as a separate *JSON* document. This document store does not allow filtering or sorting of the data without parsing each document separately. A proprietary database, instead, enables the processing of the needed user, item or user–item data in a way that is optimized for the algorithm. That is why the data are loaded at regular intervals from the content repository or the *Learning Record Store* into the recommender's database. After a specified interval (in the *Smart Learning Recommender* every 15 seconds), the *Data Access Object* layer of the *RE* calls the *Learning Record Store* in order to obtain the latest *xAPI* statements. This information is allocated to the user and item data in the recommender's database.

4.5. Conclusion of the Reference Architecture

This chapter introduced a reference architecture for adaptive learning environments with a *Recommender System* which has been applied in different educational and institutional settings. The basis is always the utilization of open standards and specifications which aim at making the whole infrastructure and particular components interoperable. Thereby, the integration of the *Recommender System* gained a special focus.

Scientific Hypothesis /SH3.0/:

An educational *Recommender System* for the recommendation of course items should build on well-selected open specifications for interfaces and data formats. If a course is based on monolithic specifications, such as the *Sharable Content Object Reference Model (SCORM)*, or on proprietary solutions, such as the XBlocks of Open edX, it would impede the utilization of a *Recommender System* for course items. Thus, the atomicity of specifications, such as given in the *Learning Tools Interoperability Specification (LTI)*, the *Learning Resource Meta-data Specification (LOM)*, the *Question and Test Interoperability Specification (QTI)*, the *Common Cartridge Specification (CC)* as well as in the *Experience API (xAPI)*, may be used for the definition of interfaces and the exchange of the required data for an educational *Recommender System.*

The next chapter discusses the issue of appropriate datasets for the evaluation of educational *Time-Aware Recommender Systems* and introduces datasets that have been collected through the use of the Smart Learning environment during teaching.

Next Chapter

5. Collected Activity Data in Field Trials

About this Chapter

Project evaluations and comparisons between different approaches should be performed using real-world conditions and data to guarantee reliable results. Nevertheless, open access datasets are not very common in the educational domain as most solutions are either not published or only partly published and/or the data are not publicly available. Within this chapter, general considerations regarding academic datasets for *Recommender Systems*, and especially for *RSs* in the area of *Technology Enhanced Learning*, are presented. A lack of open access data leads to the need to collect such data by conducting courses based on the realized architecture presented in the previous chapter. The data from three different courses are presented which subsequently build the basis for further evaluations.

5.1. Academic Datasets

The practice of using datasets from other domains than education, and in particular from the movie domain, "lacks the necessary validity for proving recommendation algorithms for *TEL*" [269, p. 3]. Reliable datasets need to "capture learner interactions in real settings" and should give the opportunity for "verification, repeatability, and comparisons of results" [269, p. 2]. These requirements are a result of the special *TEL Recommender System* paradigm introduced in Section 3.2.

> "Unfortunately, only a few datasets are published (e.g., [69] or [279]) and no data set matches all requirements of this approach. At least, the challenge data set from KDD Cup 2010 on Educational Data Mining [249] matches some of the requirements. It is divided into 5 different packages (e.g., "Algebra I" and "Bridge to Algebra" from 2005 and 2008) with between 575 and 6,043 students per package. It contains a detailed description of the students' performances when solving mathematical problems and thus, represents typical learning behavior. [...] However, the KDD dataset contains a lot of information on the interaction with *Learning Objects* as well as the processing time and results in exercises, but data on other essential factors as well as structured metadata on the hierarchy and topical sequences of *Learning Objects* are missing"

Cf. [161][72].

Lack of Academic Datasets

Because of its lack of *Learning Object* descriptions and its focus on assessments, the KDD dataset seems to be inappropriate for an evaluation of the work introduced here. Verbert et al. [269] started to analyze some further datasets for their appropriateness for the evaluation of *TEL Recommender Systems*. They described and evaluated the 6 *TEL* datasets that have been submitted to the dataTEL challenge[73]. The analyzed collections originate from different learning settings, such as a library of digital research papers [138], a work-integrated learning system [31] and a platform for open- [95] and closed-corpus contents from different educational institutions [192, 283, 275]. Thereby, the qualitative and quantitative coverage of the data differ greatly: from 6 users [31] up to 200,000 users [138] collected over periods between 3 months [31] and 3 years [283]. Unfortunately, the theoretical datasets that incorporate time information are no longer available[74]. However, even these datasets show a very high degree of sparsity, which could only partly be compensated for by incorporating implicit features (as noted by Verbert et al.) and this still results in a very low recommender accuracy of under 5% [269, p. 14-16].

Often, researchers argue that there is a lack of open, shareable datasets which incorporate contextual learning data and allow for a comparison of the results with common measurs [92, 191, 270]. Besides the missing definition of any standardized formats, there is an issue regarding privacy and legal concerns, which differs on a country by country and institution by institution basis.

5.2. Data Collection in Real-World Courses

Drachsler et al. further analyzed this gap and developed guidelines for generating suitable data for *Recommender Systems* in the area of *Technology Enhanced Learning*, which contain 3 main criteria [92, p. 2850]:

1. The dataset should "realistically reflect[...] the variables of the learning setting". The evaluation data should be, therefore, similar to the data in the real application.
2. The dataset should contain a "sufficiently large set of user profiles". Obviously, the greater the number of users, the more reliable the evaluated results, which corresponds to the law of large numbers. However, this criterion might contradict the first criterion of realistic settings. In this case, the first criterion is preferred in this work.

[72] The paper excerpt has been exclusively written by Christopher Krauss.

[73] The website "TEL Europe: dataTEL Data Set Collection" currently lists 9 *TEL* datasets (including the 6 described by Verbert et al.) – See `http://www.teleurope.eu/pg/pages/view/50630/` (Accessed: 24.07.2017).

[74] In particular, the two most promising datasets that are also listed on the TELeurope website (MACE dataset [283] by Fraunhofer FIT and Travel well dataset [275]) are not available online anymore. The responsible persons have been contacted via e-mail, but, in the case of the MACE dataset, the responsible staff changed employer.

3. The dataset should be "comparable to others". The criterion of comparability comprises quantitative aspects (e.g., the number of learners, *Learning Objects* and activity data) as well as qualitative considerations regarding the course setting (face-to-face, online, blended learning, etc.), the didactic structure of topics and the format of the *Learning Objects*.

On the one hand, the recommendation tasks should be "similar to the tasks supported by the system from which the data was collected" [125, p. 15]. On the other hand, researchers should "adequately define the reference variables against which the adaptivity of the system will be evaluated" [191, p. 404]. Thus, it would be inappropriate to use data from a different domain or from services that do not focus on the same use case. Verbert et al. argued that "the continuation of additional small-scale experiments with a limit amount of learners that rate the relevance of suggested resources only adds little contributions to an evidence driven knowledge base on *Recommender Systems* in *TEL*" [269, p. 16].

5.2.1. Energy-Consultant Training

The Smart Learning environment was initially designed to fit the needs for an Energy-Consultant Training at the Chamber of Crafts. Thereby, the *Learning Companion Application* was utilized as learner interface[75]. Consequently, the system is evaluated with participants and institutional staff of the Chamber of Crafts.

> "In Germany, the Chamber of Crafts (Handwerkskammer [Berlin]) provides numerous vocational training that leads to the obtainment of a certificate. Trainees are full-time professionals. Until now most of the training is fully face-to-face. The aim of the project 'Smart Learning in Vocational Training' [168] is to introduce a blended-learning approach in the training of Energy Consultants. Learning material is currently structured and developed using different digital media: texts, animations, screencasts, videos. During lecture phases, trainees learn hands-on with a professional"
>
> Cf. [166, p. 472][76].

The first course run at the Chamber of Crafts Berlin was conducted between September and December 2016 with eight participants. A second course run took place between March 2017 and July 2017 with six participants and a third course ran from September to December 2017 with another eight participants. The master craftsmen enrolled on the course partly funded this training in order to receive the permission to issue energy efficiency certificates. It is a Blended Learning course with lectures in the evening as well as on weekend and a final exam at the end

[75] As mentioned in the last chapter the visualization of *LCA* is introduced in Appendix B.23.
[76] The text has been written by Prof. Dr. Agathe Merceron. Martin Dinziol was responsible for the course runs at the Chamber of Crafts.

that is managed by another institution. The five trainers presented six topics in 14 face-to-face lectures per course run. In the first course-run, the online material included 14 sub-topics (called *Learning Units*) with about 50 *Learning Objects* as a mixture of texts, videos, animations and quizzes [77]. In the third course-run, the Chamber of Crafts presented about 366 *Learning Objects* (including automatically generated *LOs*, such as self-assessments on learning objectives) in 39 *Learning Units*.

5.2.2. JavaFX Online Course

In addition to the traditional Blended Learning course at the Chamber of Crafts Berlin the Smart Learning environment has been utilized for an online-only course that is offered in addition to the mandatory courses at the Beuth University Berlin.

> "The JavaFX online course, available for a period of 11 weeks, offers an introduction into the FX-Framework for the development of platform-independent Java applications and targets bachelor computer science students. [...] By taking part in this course, a student did not earn any mark for her/his studies, only knowledge for herself/himself. In this sense, JavaFX is similar to a *MOOC* but without the massive and open part.
>
> It comprises three *Learning Units*. Each *Learning Unit* has about 5 learning objectives and contains about fifteen to thirty *LOs* (units are not of equal length). About half of the *LOs* are texts to explain concepts and exemplary programs, and half are exercises (single/multiple choice, cloze and so on). The last *LO* before the self-assessment of the learning objectives is a comprehensive programming task; students can send their program per email and obtain a manually commented evaluation. Based on the educational discussion on *MOOCs*, Daniel pointed out that 'students seek not merely access, but access to success' [77]. However, success can be different for each student. Driven by this consideration, a specific *LO* has been added to this course allowing each student to rate her/his motivation on a scale from 0 (do nothing) to 100 (engage thoroughly with everything offered). 51 students enrolled in this course, however, there were 23 no-shows – people register but never log in to the course while it is active. Only the remaining 28 students are considered for the analysis in this paper. The 28 users generated 3624 *xAPI* statements in total during the course"
>
> Cf. [17][78].

Appendix B.26 contains an activity overview for the whole period of the 11 course weeks. In

[77] Appendix B.25 gives an overview of the approximately 3,000 *xAPI* statements collected per course. Most of the activities occurred at the end of the course, right before the final exam. All participants passed the exam. However, marks are unknown due to privacy issues relating to the external examinations board.

[78] This paper excerpt was primarily written by Prof. Dr. Agathe Merceron and Sinh Truong-An who also managed the course.

contrast to the massive learning phase at the end of the Energy-Consultant Training, the JavaFX course shows a more scattered distribution of learning activities over the whole course period.

5.2.3. Advanced Web Technologies Course

The *FOKUS* research engineers under the lead of Dr. Stephan Steglich offer a standard yearly course on *Advanced Web Technologies (AWT)* at the *Technische Universität Berlin (TU Berlin)*. On completion of the lectures, students earn 3 credits. In its first iteration, the lecture slides were offered on Moodle[79]. For the second and third course runs, the course offered digital media via the *Learning Companion Application*.

First Moodle Course

> "*Advanced Web Technologies (AWT)* targets master computer science students. Technical experts [teach] in 12 weekly presence lectures diverse topics that are of interest for future web developers – from web technology basics, such as *HTML*, over media delivery and content protection, to personalization through *Recommender Systems* and the Internet of Things. The lectures are mostly held with slides created in PowerPoint showing definitions, specifications, and source code, animations for concepts and videos for practical examples"
>
> Cf. [17][80].

The first course run in the winter semester of 2015/2016 did not offer any adaptive functions, there were no Learning Analytics for the instructors[81] and no educational *Recommender System* for the learners. The uploaded materials consisted exclusively of files in the *Portable Document Format (PDF)* with copies of the presented PowerPoint slides for each lecture – 12 files in total. Topics were presented weekly, and each lecture slide set was uploaded as a separate file after the lecture.

> "At the end of the course, students can earn credits by completing a one-hour exam consisting of 50 multiple choice questions and 5 bonus questions. [...] note that the best mark is 1.0 and the worst possible mark is 5.0, which means failing the exam [...]"
>
> Cf. [17][82].

In this first course run, 39 (of 72) students participated in the final exam (1 failed in the first attempt and repeated the exam successfully two months later). The average mark was 2.00, which

[79] The *TU Berlin* instance of Moodle is called *Information System for Instructors and Students (ISIS)*. See `https://isis.tu-berlin.de/` (Accessed: 07.07.2017).

[80] This paper excerpt was primarily written by Christopher Krauss. The course is hosted by Dr. Stephan Steglich with the support of 4 scientific employees of the *Fraunhofer-Institute for Open Communication Systems* – among whom Christopher Krauss additionally managed the *LCA* course offering.

[81] Although Learning Analytics were theoretically provided by Moodle, the instructors did not have access to Moodle.

[82] This paper excerpt was primarily written by the author of this dissertation.

corresponds to a success rate of 84% in multiple choice tests – see Table 5.1. This course run serves as the baseline without adaptive features.

Smart Learning Course

For a second iteration of the course, the instructors used the Smart Learning environment to present lecture materials, a number of quizzes and additional materials. See Figure 5.1 for the course outline of *Advanced Web Technologies* rendered in the *Learning Companion Application*.

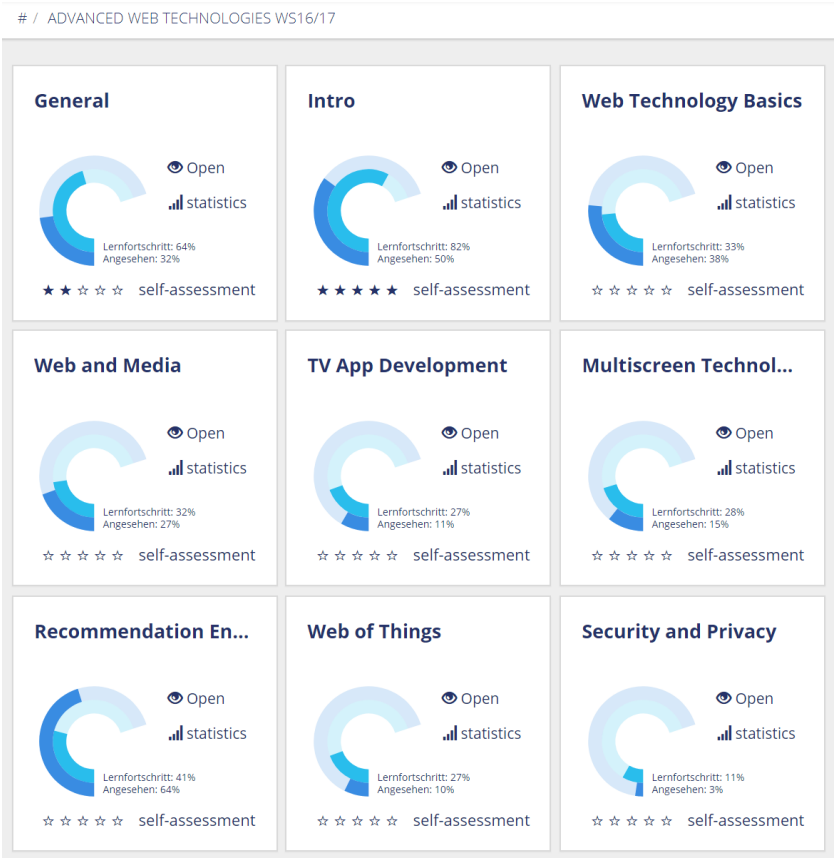

Figure 5.1.: Screenshot of the *AWT* course outline presented in a version of the *LCA* on a desktop browser that was translated into English via a third party component.

> "The about 1000 presented slides are converted to digital *Learning Objects*, one slide being a single *LO*, and grouped into [106] *Learning Units* for the representation in *LCA* – with videos, animations and additional multiple-choice questions at the end of the *Learning Units*. Moreover, as some students still want to learn with a printed version of the slides, the last *LO* of the accordion view is a *PDF* file to be downloaded that contains all the slides of the unit"
>
> Cf. [17][83].

[83] This paper excerpt was exclusively written by Christopher Krauss.

xAPI Statements

The 92,291 mined *xAPI* statements of 99 active users in the second course iteration[84] with *LCA* are compounded as follows (for a description of the *xAPI* verbs used see Section 4.3.3):

1. 44421 "initialized" *xAPI* statements
2. 29712 "exited" *xAPI* statements
3. 8384 "answered" *xAPI* statements
4. 7636 "abandoned" *xAPI* statements
5. 1019 "downloaded" *xAPI* statements
6. 861 "waived" *xAPI* statements
7. 81 "suspended" *xAPI* statements
8. 72 "commented" *xAPI* statements
9. 63 "resumed" *xAPI* statements
10. 42 "completed" *xAPI* statements

Figure 5.2 shows a visualization of the statements produced per day over the period of the course from October 24, 2016, until February 12, 2017 (the first lecture was held on October 27, 2016, and the final exam was on February 09, 2017). Apparently, most of the statements were made at the end of the course [85].

Figure 5.2.: Learning Locker Activity Chart for AWT for the whole course period (Number of xAPI Statements per day in blue); x-axis: days of the course; y-axis number of *xAPI* statements.

Test Results

Of the 118 students enrolled for *AWT* in the winter semester 2016/17, only 99 of them used the *Learning Companion Application*.

> "Especially in the first weeks before the official registration deadline, students frequently change their mind for participating in specific courses – which might explain the high loss ratio of the participants. [...] Exactly 75 students completed the final exam (even two who did not use the *LCA*) and the average mark was 1.90 [...]"
>
> Cf. [17][86].

[84] Unfortunately, the third course run did not end before the submission of the dissertation. However, by the end of the enrollment deadline in November 2017, 116 students registered for the *Learning Companion Application*.

[85] Appendix B.27 contains a visualization of the activities without the statements from the final 3 weeks of the course. This figure indicates that students are, first and foremost, working with the *LCA* during the lectures.

Eight additional students also participated in the exam but without official enrollment. Thus, for them, the assessment had no official impact. See Table 5.1 for the distribution of marks for all 83 students – unfortunately, there is no information regarding who of the 83 students did not enroll but did participate in the exam.

Table 5.1.: Average Marks per student in AWT in winter semester 15/16 and winter semester 16/17

Correctly answered questions	Mark	Students ws15/16 (in %)	Students ws16/17(in %)
> 95%	1.0	9 (23.1%)	16 (19.3%)
90 - 95%	1.3	2 (05.1%)	21 (25.3%)
85 - 90%	1.7	12 (30.8%)	9 (10.8%)
80 - 85%	2.0	3 (07.7%)	8 (09.6%)
75 - 80%	2.3	6 (15.4%)	6 (07.2%)
70 - 75%	2.7	3 (07.7%)	10 (12.0%)
65 - 70%	3.0	1 (02.6%)	3 (03.6%)
60 - 65%	3.3	0 (00.0%)	7 (08.4%)
55 - 60%	3.7	2 (05.1%)	1 (01.2%)
50 - 55%	4.0	0 (00.0%)	0 (00.0%)
< 50%	5.0	1 (02.1%)	2 (02.4%)
Average Mark		2.00	1.90
Exam Participation		39	75 (+ 8)
Course Enrollments		72	118 (+ 8)
Drop-Out Rate		46%	36% (34%)

The analysis of the exam results indicates that the direct conversion of lecture materials that were presented as digital media had no, or just a small effect, on exam performance[87]. However, the total number of participants and the drop-out rate improved in the second iteration with *LCA* and including the recommendations of the *Smart Learning Recommender*. The mined usage data allows for a further analysis of the learning activities and yields a deep understanding of different learning patterns and the impact of each. This knowledge, in turn, is utilized to make learning more efficient and help in the attainment of the course goals. This is why the datasets are used for the evaluation.

[86]This paper excerpt was primarily written by Christopher Krauss.
[87]Due to the number of participants, the improvement of the average mark of 0.1 is not significant.

5.3. Composition of the Collected Datasets

It is common sense that evaluations of *Recommender Systems* focus on the prediction of the future behavior of individual users [147, 84, 33]. Thereby, the actual activity data collected in the past are compared with the predicted activity data. Transferred into the *TEL* domain this means that, independent of the learner's success, the algorithms aim at forecasting the next item for consumption.

Learning Objectives

This approach lacks some critical considerations regarding the special *TEL Recommender System* paradigm. Thus, an inactive learner would, for instance, only receive recommendations based on the activities of other infrequent learners, but unfortunately not the needed recommendations that help to pass the final exam.

Learning Success

Regarding motivation or the probability of success on the course, learners should be given recommendations that help to reach the course goal (e.g., pass the final exam). Therefore, different learner types and the success of each must be differentiated. Consequently, a *Collaborative Filtering* approach can be extended to take into account only the activities of successful learners – even for in respect of the recommendations given to weak learners. While an accuracy evaluation based on this past data would show worse results, learners benefit from more stimulating predictions that leverage their potential learning efficiency.

This Section

This section focus on the identification of different engagement patterns of the user groups and their overall course success in the collected datasets. Thereby, distinct clustering approaches are applied and evaluated. The main aim is to identify successful learners and to transfer their learning patterns to users at risk of failing the course goal such as weak users and even potential drop-out users to improve their learning efficiency.

AWT Activities

A lot of research has focused on the identification of typical learning patterns [156, 204, 155, 109, 17][88]. The *LCA* users also show huge discrepancies in the usage of the app. This can be observed between the different course settings as well as between different users[89].

> "[For the *AWT* course, a] total amount of 84% of all interactions on *LCA* were done on Desktop computers, the rest on mobile devices. In contrast to [the Energy-Consultant Training] performed with *LCA* at a Chamber of Crafts where almost 40% of all interactions were performed using a smartphone or tablet, the *AWT* usage pattern surprisingly did not correspond to the paradigm of mobile learning at smartphones or tablets. [...]
>
> *LCA* is most frequently used during the working hours from 8:00 am to 6:00 pm – with two major exceptions: Computer science students learn a lot during noon (especially Thursdays during the lecture times 12 pm – 2 pm) and they start interacting with the app again in the

[88] A section on the related work of typical learning patterns can be found in Appendix B.28.
[89] See Appendix B.29 for a figure presenting the statistics of the following paper excerpt.

evening between 7 pm and 1 am"

Cf. [169, p. 3][90].

Figure 5.3 visualizes the number of *Learning Unit* accesses per student in the *Advanced Web Technologies* course – separated by first visits (blue) and repetitions (orange). The total number of unique *Learning Units* is 106 while a single student accessed only 89 *Learning Units* at most. Figure 5.4 displays the increasing number of average *LU* accesses per week. Among the rarely consumed items are "Introduction", "Module Information" and "Motivation" which involve the *Learning Units* presenting a general introduction to the course and which are not relevant for the final exam.

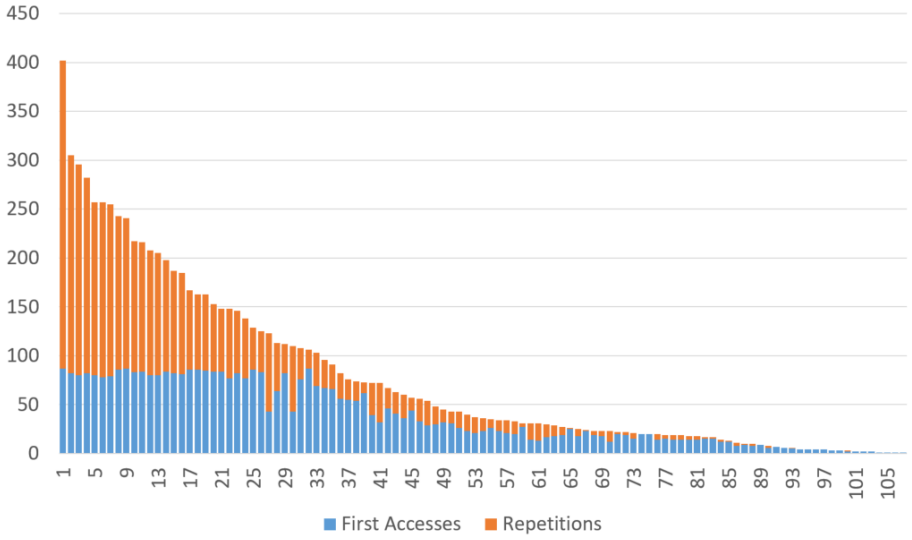

Figure 5.3.: Number of accesses of *Learning Units* (y-axis) per student (x-axis) via the *Learning Companion Application* in *Advanced Web Technologies*.

As one can see, the *AWT* learners showed different patterns of usage. In general, the greater the number of item accesses, the more items have been repeated. In contrast to this, students who studied less than 100 topics only repeated a low percentage of them and users with less than 20 accesses repeated almost no items. However, there is also an unknown number of cases where users downloaded the *PDF* and learned offline. Thus, the analysis can only be seen as evaluation of an significant part of the learning activities – not on the entire learning progress of all learners.

Clustering Technique

The learners have additionally been clustered with the idea of grouping the students based on their interactions with the *Learning Companion Application* that are stored as *xAPI* statements.

"The collected *xAPI* statements [in the *AWT* course] (on elapsed time per *LO* and assessment scores in exercises) have been processed in order to discover typical learning patterns [17].

[90] The analysis has been made and the paper excerpt was written primarily by Christopher Krauss.

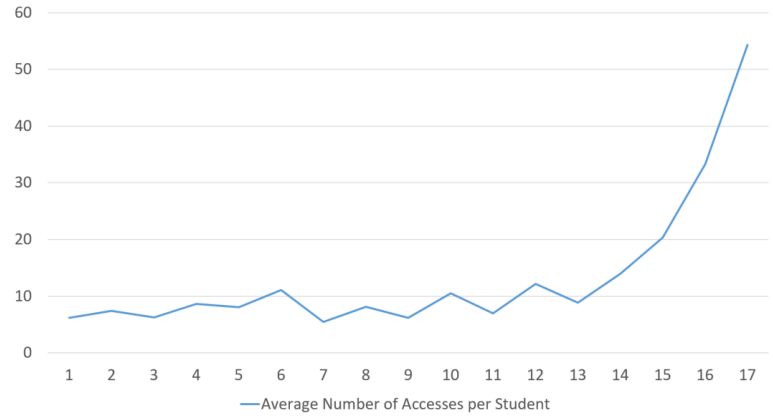

Figure 5.4.: Average number of accesses of *Learning Units* (y-axis) per week (x-axis) via the *Learning Companion Application* in *Advanced Web Technologies*.

> We identified 3 clusters, labeled as Completers, Strong Starters and Auditors first introduced by Kizilcec [155]. Subsequently, we calculated the average mark in the final test per cluster. The Completers' cluster contains the 26 students who accessed all *LOs* and solved correctly all exercises. Their average mark of 1.5 is better than the total average of 1.9. The 9 Strong Starters had an average mark of 2.0 (almost the total average). At a certain point in the course, Strong Starters engaged less with *LCA* than Completers. The Auditors' cluster, in contrast, shows students who infrequently used *LCA* and is the biggest cluster with 64 students – including all students who did not participate in the final test. They reached an average mark of 2.2 which is less good than the total average"
>
> Cf. [169, p. 3][91].

While the clustering shows reasonably typical learning behaviors at the end of the *AWT* course, this approach cannot be directly transferred to educational *Recommender Systems*. The *RS* requires information regarding favorable patterns right from the beginning of the course as the algorithms needs to incorporate the information of successful patterns even for the cold start phase. Thus, an advantageous behavior needs to be taken from comparable course settings, e.g., a previous course run. More importantly, the classification of current course participants is required at every point in time during the course period. Unfortunately, this is very challenging because typical learning clusters are not built until a specific duration has elapsed, particularly for "Nearly There" users or "Late Completers" [109].

KNN Prediction

A *K-Nearest-Neighbour (KNN)* algorithm with cosine similarity on "initialized" *xAPI* statements has been applied to predict the final mark through the identification of k similar learners in the

[91] The paper excerpt has primarily been written by Christopher Krauss. However, the approach of the clustering of learners with K-Means and X-Means was initialized by Prof. Dr. Agathe Merceron and her team of the Beuth University. Christopher Krauss contributed with data of the *AWT* course and the success analysis.

LCA[92]. In contrast to K-Means, where the result is a fixed amount of clusters and each user is grouped into just one cluster, the *KNN* approach creates groups of similar users around the requesting user. That means that each cluster contains the same number k of learners and learners can appear in multiple clusters. The algorithm performs best with a setting of 10 to 30 neighbors (which means about 10 to 30% of users). The prediction has an error[93] of 0.8 (on a scale of 1.0 to 5.0) over the whole course period so that the grade of all k neighbors deviates from the actual user's degree by about 0.8 on average[94].

In the last six weeks of the course, smaller values of k (3 to 4) show, however, slightly lower error values of about 0.75. Moreover, after the final exam, when all users stopped accessing the learning materials, the setting of $k = 1$ (only one other user is considered as a neighbor) showed an error of 0.68. As a conclusion, the longer the course ran, the fewer the number of classmates needed to produce more accurate grade predictions. However, the *KNN* approach allows only rough forecasts of grades during the run of a course and requires historical data from the same course setting. Thus, it cannot be utilized as an alternative to the clustering approach.

5.4. Conclusions of the Collected Datasets

This chapter described the mined activity data relating to three courses that will enable the analysis of different educational *Recommender Systems*.

Scientific Hypothesis /SH4.0/:

The activity data collected during the conducted courses, especially the dataset relating to the *Advanced Web Technologies* course, may be utilized to evaluate closed-corpus *Recommender Systems* for the prediction of course items.

Next Chapter

The next chapter presents the methodological approach for this dissertation which is the basis for further realization and evaluation steps. Thus, the evaluations build on the collected activity data.

[92] Related work of course grade predictions can be found in Appendix B.31. The *TU Berlin* student Alexandru Ciobanu participated in the module "Project *Advanced Web Technologies*" with the project "K-Nearest-Neighbour-Algorithm for Prediction of Course Grades" supervised by Christopher Krauss. The concept were provided by the supervisor, while Alexandru Ciobanu realized the implementation and evaluation.

[93] The utilized *Root Mean Square Error (RMSE)* is introduced in the next chapter.

[94] When the average course grade of 1.9 is used as prediction value (baseline), the forecast performs only slightly worse and has an error of about 0.9.

6. Methodology and Evaluation Design

Evaluation is the "systematic investigation of the worth or merit of an object" [229, p. 3]. In more detail, it is "the identification, clarification, and application of defensible criteria to determine an evaluation object's value, quality, utility, effectiveness, or significance in relation to those criteria" [285, p. 5] (see also [191, p. 404]). This work respects and follows Balzert's science ethics [28, p. 13-15] comprising "relevance", "orginality", "honesty", "fairness", "understandability", "reliability", "objectivity", "validity", "logical reasoning" and "confirmability" of scientific evaluations[95].

About this Chapter

As introduced in the previous chapter, the experiments of this dissertation start with the conduction of *TEL* courses and a collection of learner activities relating to the content. The methodology diagram is presented in Figure 6.1. The collected data need to be normalized and, if necessary, also cleansed. For instance, teachers and other institutional staff need to be removed from the learners' activity data. Based on the clean data, a recommendation model is applied that predicts the item's relevance. The relevance scores are used for an internal ordering of the items and this process results in a Top-N recommendation list. In the end, the recommendations are evaluated, and conclusions are drawn, to optimize the model. This chapter introduces the general evaluation procedure. The next chapters, in turn, focus on the algorithm design, approximation of the relevance scores, the prediction of recommendations and its evaluation. The evaluation results are, then, utilized for proving or disproving the hypotheses[96]. Finally, the models are optimized in an iterative process.

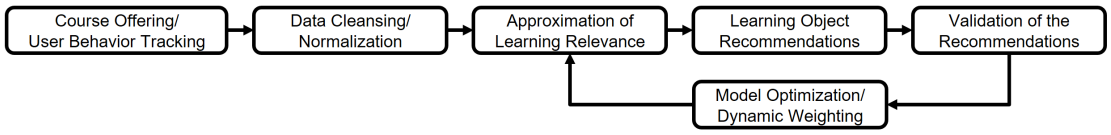

Figure 6.1.: Methodology diagram starting with the course offering to the iterative point where the recommendations are optimized.

[95] The list of criteria, which will be the basis for further research and experiments, and the meaning of the latter in the context of this dissertation are presented in Appendix B.32.
[96] Appendix B.33 presents an introduction of qualitative and quantitative hypothesis reasoning.

6.1. Evaluation Framework

Researchers have noted that besides a standardized dataset definition, there is also a lack of standardized definitions relating to evaluation procedures for *Recommender Systems* in *Technology Enhanced Learning* [94, 92, 191]. They suggest approaches but also comment that they must be further researched.

The most important aim of an evaluation "is to measure a certain property or effect of the *Recommender System*" [103, p. 327]. Said and Bellogin evaluated common evaluation frameworks and protocols. They conclude that the "performance of an algorithm implemented in one [evaluation framework] cannot be compared to the performance of the same algorithm in another" [228, p. 135]. Their results differ by up to 10% depending on the evaluation framework (LensKit[97], Mahout[98] and MyMediaLite[99]). Moreover, there "are no de facto rules or standards on how to evaluate a recommendation algorithm" [228, p. 135].

Chatti et al. argued that "an implementation of different recommendation algorithms within a single *Recommender System* to compare against each other is missing in the *TEL* recommenders literature" [58]. Thereby, "further evaluation procedures that complement the technical evaluation approaches" for the comparison of educational *Recommender Systems* are needed to produce reliable and comparable results [191, p. 22]. Campos et al. noticed that the existence of a huge variety of evaluation approaches for general *Recommender Systems* results in "an increasing impediment to fairly compare results and conclusions reported in different studies" [57] (see also [33]). Moreover, "variations in user interfaces", "data selection" and "situational and personal characteristics of users" lead to differences between qualitative and quantitative evaluations [57, p. 83]. This is why the underlying evaluation setting must remain the same in each experiment.

Evaluation Protocol

Weibelzahl introduced a framework for a four-tiered evaluation procedure [281, p. 50-55] consisting of "evaluation of input data", "evaluation of the inference mechanism", "evaluation of the adaption decision" and "evaluation of the total interaction". Manouselis et al. abstracted this to a multi-layered evaluation approach for *RSs* in *TEL* which can also be reduced to only two layers: the accuracy of the model and the effectiveness of the changes made at the interface [191, p. 405]. The first layer corresponds to quantitative evaluations, relating to measurements of the algorithm's outputs, the second to qualitative assessments regarding user perceptions. This work has, similar to other *RS* evaluation procedures, a particular focus on quantitative aspects. However, qualitative aspects are evaluated with the course participants[100], but due to the sample size representativeness is not claimed.

According to Said and Bellogin [228], the most important evaluation dimensions are (1) the

[97] LensKit. See: `http://lenskit.org/documentation/evaluator/` (Accessed: 26.11.2017).
[98] Mahout. See: `https://mahout.apache.org/` (Accessed: 26.11.2017).
[99] MyMediaLite. See: `http://www.mymedialite.net/` (Accessed: 26.11.2017).
[100] An introduction to the qualitative evaluation procedures is presented in Appendix B.34.

dataset, (2) the data splitting, (3) the evaluation strategies and (4) the metrics (here called measurement values). This work builds on these four dimensions and follows additionally the approach of Campos et al. [57] who suggest that the following questions should be answered during each experiment:

- "MQ1: What base set is used to perform the training-test splitting?
- MQ2: What [scoring] order is used to assign [relevance scores] to the training and test sets?
- MQ3: How [much data] comprise the training and test sets?
- MQ4: What cross-validation method is used to increase the generalization of the evaluation results?
- MQ5: Which items are considered as target items (in a Top-N recommendation task)?
- MQ6: Which items are considered relevant for each user (in a Top-N recommendation task)?" [57, p. 87]

Simulated Students

The required evaluations can be either performed online – directly in a real course – or offline — with the use of a simulation. The latter can utilize either simulated data or past real-world data in a simulated environment [57, p. 74].

Since it is cost intensive to conduct experiments in real learning environments "most authors exploit simulated single skill students based on different technologies like *Artificial Neural Networks* or self-developed student models [233], [188]" [237]. That is why Schatten and Schmidt-Thieme used a "student simulator that partially overcomes the problem of massive testing with real students" [237]. They first identified different rules of typical learners (e.g., "The total knowledge at the beginning is different than zero") and the learning contents (e.g., "A content is either of the correct difficulty for a student, or too easy, or too difficult"). These definitions were strictly followed when generating random data that could be used for exhaustive evaluations. However, the weakness comes from the unpredictability of real human behavior. A single person, as well as a whole community, does not act in a random manner, they act in complex patterns. If today researchers were able to model these complex patterns, the key task of a *Recommender System* would have been solved. Also, Campos et al. note, that "the majority of past work on *TARS* [...] has been focused on offline evaluation protocols" [57, p. 74].

In this dissertation, a mixture of real and simulated evaluation procedures is applied. Each experiment incorporates the activity data that have been collected in real-world courses (as presented in Chapter 5). However, the evaluations are not performed online in a live course setting, but rather offline using past data. Thus, the evaluated recommender algorithm does not influence the future behaviors of the learners as a live evaluation would do. It aims at forecasting the usage patterns that are already present in the data. This mixture is very common for evaluations of general *Recommender Systems* [197, 57] (and even for the Netflix prize [34]) and has been also applied for *TEL Recommender Systems* [191, 269, 103].

6.2. Data Splitting and Cross-Validation

Data Splitting

When performing an offline evaluation with historical data, the whole dataset must be split: To guarantee an objective prediction of data, the training dataset Tr must be separated from the test dataset Te [125] [57, p. 74]:

$$Tr \cap Te = \emptyset. \tag{6.1}$$

It is common sense that the split process happens randomly. With one restriction: at least some consumption data of the same user u, who will receive recommendations, should exist in Tr and Te. If there are no data of user u in the training set Tr, a *CF* algorithm would fail because of the cold start problem. More importantly, if there are no activity data available in the test set Te, this user cannot be evaluated since a prediction cannot be compared with real data. Thus, either the prediction of a relevance score or the evaluation of this prediction would be impossible.

> "One method to train and evaluate algorithms is the use of the n-fold cross-validation. This cross validation type is repeated ten times [for $n = 10$]. During every iteration the whole data set is split into another 10% [(corresponds to $\frac{1}{n}$)] of evaluation data and 90% of training data [(corresponds to $\frac{n-1}{n}$)]. The average value of errors defines the accuracy"
>
> Cf. [167, p. 67][101].

Until now, *TEL Recommender Systems* have only been evaluated, if at all, through the standard cross-validation setting [191, 269][102]. However, this contradicts the specialized *RS* paradigm in *Technology Enhanced Learning* where time plays a key role (as introduced in Section 3.2).

Validation in TARS

The evaluation of *Time-Aware Recommender Systems* shows an additional restriction: To simulate real-world behavior, the split of Tr and Te is not entirely random, but depends on a particular point in time – which might be chosen randomly in a given time interval [57, p. 75]. This second restriction makes it unfair to compare *TARSs* and time-unaware *Recommender Systems*, as the latter are mostly evaluated without taking into account time information for splitting.

Lu et al. argued that "most *Collaborative Filtering* algorithms have been designed and evaluated based on random split of training and test sets without considering the temporal structure" [185, p. 13]. Therefore, Campos et al. suggest different specialized validation approaches for *Time-Aware Recommender Systems*, where only two definitions seem to be appropriate for the evaluation of learning environments [57, p. 94]. The "increasing-time window" approach splits the whole dataset Tr and Te according to a variable time threshold $thresh$ so that all data in Tr are older than $thresh$ and all data in Te are younger. Of course, the time threshold needs to be set within a

[101] The section about cross-validation was written by Christopher Krauss. However, the concepts are well known (e.g., referenced in [125] and [33, p. 335]).

[102] While a time-based evaluation for *TEL Recommender Systems* was also mentioned in a survey by Erdt et al. [103, p. 328], there is no information whether this approach has been applied to *TEL RS*.

reasonable interval, e.g., for a course, between the start of the course $tCouseStart$ and the end of the course $tCourseEnd$, so that Tr and Te are not empty. Appendix B.35 presents an example of the increasing time-window cross-validation.

Another approach is called the "fixed time-window" approach that works in a similar manner to the "increasing time-window" approach, but using a fixed time interval int for both the training dataset and the prediction dataset. In a course, the threshold $thresh$ is still variable, but the data in Tr are restricted. The time of each training data point t_{Tr} must to be in the range of $thresh - int < t_{Tr} < thresh$ and the time of each test data point t_{Te} must be in $thresh < t_{Te} < thresh + int$.

Yi et al. [295, p. 119-120] conducted an implicit fixed time-window cross-validation for a search engine evaluation task (without stating it as such). Thereby, the authors analyzed common measures for different sizes of the time window: one month, one week and one day. The dataset composition of the one-month setting is unique, as only the final month data were used as test data and the consumption data before this time were used as training data. Interestingly, some measures (e.g., mean absolute precision) were 40 % better for the weekly time windows compared with the monthly time window. The weekly and daily settings, in contrast, showed almost similar results. In conclusion, it is clear that researchers must consider that for a *Time-Aware Recommender System* evaluation because the selection of the time-window size and splitting approach might also influence the evaluation results. However, time-window evaluations are not comparable to standard cross-validation evaluations.

6.3. Algorithmic Measurements of Appropriate Recommendations

Appropriate Recommendations

A critical question regarding evaluation is: how to measure "appropriate" or "good" recommendations. Campos et al. pointed out that there is no definition of what constitutes a "good" recommendation, but "a commonly used approach is to establish the quality (goodness) of recommendations by computing different measures that assess various desired characteristics of an *RS* output" [57, p. 74]. Manouselis et al. recommends four high-level measures to define success criteria of *Recommender Systems* in *TEL* [191, p. 406]:

1. **Effectiveness** describes the percentage of consumed *Learning Objects* during a learning phase (here a course).
2. **Efficiency** indicates the time needed by the user to reach the learning goal.
3. **Satisfaction** is a subjective measure that must be assessed by discussion with users.
4. **Drop-out rates** represent the percentage of users who stop participating in the learning setting and thus do not reach the course goals.

Erdt et al. [103] classified similar measures in "*Recommender System* Performance", "User-Centric Effects" and "Effects on Learning". According to them, popular performance measures in offline experiments are *Mean Absolute Error*, *Root Mean Square Error*, precision, recall and f-score.

Moreover, Manouselis et al. [191] incorporated some further measures from *Social Network Analysis*, such as Variety, Centrality, Closeness and Cohesion, as they seem also to be valid for learning networks. Due to the course setting of the collected datasets and by following the considerations of Rada [220], this work has a special focus on efficiency and effectiveness – however, results regarding satisfaction levels and drop-out rates are also presented. Such measures cannot be determined by the use of only qualitative or quantitative analysis. According to Bellogin et al. [33], each measure itself is insufficient for a fair comparison of different approaches. For instance, "putting more relevant items in the top-N is more important for real recommendation effectiveness than being accurate with predicted rating values" [33, p. 5].

6.3.1. Prediction Accuracy – The Effectiveness of a Recommender System

The effectiveness of a *Recommender System* mostly refers to its prediction accuracy [125][103]. Thereby, measurements focus on the accuracy of the predicted relevance score – for instance through a value often presented as the error [235] or through the precision of the Top-N list [84]. Both approaches are introduced in the following.

6.3.1.1. Error Measurements

MAE & RMSE

Measures to determine the prediction errors of the underlying relevance score, such as *Mean Absolute Error* and *Root Mean Square Error* are commonly applied in the *RS* domain [57, p. 85], [125]. Remember, the relevance score depends on the algorithm used[104]. In *TEL* a relevance score can be a traditional rating, a predicted numerical value representing the knowledge level or learning need, the number of *LO* accesses or even a Boolean value indicating whether an item has been consumed or not. The easiest way to calculate the error in the predicted score of an item i for a particular user u is to subtract the real value (e.g., given as a rating by the user) from the prediction value.

"The *Mean Absolute Error (MAE)* and the *Root Mean Square Error (RMSE)* are calculated as

$$MAE = \frac{\sum_{i=1}^{N} |p_i - q_i|}{N} \tag{6.2}$$

[103] An analysis of additional measures (that are not part of the evaluation framework employed here) is presented in Appendix B.36.

[104] *TEL* relevance scores have been introduced in Section 3.2.3 and can be classified as one of the attribute types introduced in Section 2.2.1.1.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} |p_i - q_i|^2}{N}} \tag{6.3}$$

$< p_i, q_i >$ is a prediction-real-value pair of N where p_i is the predicted value and q_i is the real value for item i. The lower the error the better the algorithm. Formally, $RMSE \geq MAE$ [as long as the relevance score is 1 or higher]. The greater the difference between MAE and $RMSE$ the more scattered errors are"

Cf. [167, p. 67][105].

Accuracy

MAE and *RMSE* are used to evaluate the prediction accuracy of the underlying relevance score. Given the example of a star-based *Recommender System* which collects and predicts item preferences on a one-to-five star scale. For each item rating $r_{u,i}$ within the test set Te, it is possible to evaluate a rating prediction for the same item i. Based on the data in the training set Tr, and in the absence of any knowledge regarding the real rating in the test set q_i, a rating prediction p_i is calculated. The average deviation between p_i and q_i represents the average error. The error values are in the range of the evaluated relevance score [117, p. 2944] which can either be binary, stating whether a recommendation was successful or not, or, more frequently, in the range of the predicted score or rating. To normalize errors to a relative scale (for instance to compare a *RS* using a five star scale with one that presents recommendations on a percentage scale), the accuracy value has been introduced [84, p. 792]:

$$accuracy = \frac{max(p) - min(p) - MAE}{max(p) - min(p)}, \tag{6.4}$$

where $max(p)$ corresponds to the maximum value that p (or q) can take and $min(p)$ to the minimum value. This is why the *accuracy* value is given in the range [0;1] which allows for a comparison even of recommenders that use different scoring scales.

A weakness when using accuracy measures for evaluating *TEL RSs* is the lack of comparability. While error measures are appropriate to compare deviations between predictions and the actual given relevance scores, they can be only applied for the same type of scoring approach. For instance, an error for a rating-based algorithm (from one to five stars) should not be compared to the error of a knowledge-level-based algorithm (with relevance scores given in percent). Due to the different meanings and ranges of the relevance scores, the resulting errors of the algorithms differ in their meaning, as well. For example: Is a *RMSE* of 1.0 in a rating-based algorithm worse or better than a *RMSE* of 20.0 in a knowledge-based algorithm (given in percent)? Moreover, errors do not reveal anything about the appropriateness of the resulting recommendations for a specific learner nor do they allow for a comparison of different algorithms that are based on

[105] The section concerning *MAE* and *Root Mean Square Error* was written by Christopher Krauss. However, concepts discussed within it come from [235, p. 290] and [298, p. 63] and are referenced in the paper as such.

different scoring values. Cremonesi et al. note that "improvements in *RMSE* often do not translate into accuracy improvements" [72, p. 39]. Taking these points into consideration, the following measures fit better when comparing algorithms in the areas of *Time-Aware Recommender Systems* and *Technology Enhanced Learning*.

6.3.1.2. Ranking Precision

Confusion Matrix

Campos et al. argue that the measure "ranking precision" is more appropriate for *Recommender System* tasks than error measures, as the former represent the coverage of relevant recommendations within the presented Top-N list, which are typically given as precision or recall [57, p. 85]. Del Olmo and Gaudioso [84, p. 793] introduce a confusion matrix to explain the possible states which a recommendation might have:

Table 6.1.: Confusion matrix for item classification

	Relevant	Non-relevant
Recommended	a	b
Non-recommended	c	d

The confusion matrix shows the 4 item state categories: they can either be recommended (a & b) or not (c & d) and at the same time be relevant (a & c) or not (b & d). Here, the term "relevant" comes from the Information Retrieval domain and represents "consumed items" in the case of a *Recommender System*. As the *TEL RS* aims at also predicting the future *LO* consumption, a "relevant" and "recommended" item is an *LO* that has been accessed (here represented as "initialized" *xAPI* statement) by the learner after it has been recommended. Thus, a relevant item for user u is an item that has been accessed by the same user in the test set.

Precision

Precision is the portion of recommended relevant items in the set of all recommendations (cf. [274, p. 12], [84, p. 793], [117, p. 2945], [33, p. 334]):

$$precision = \frac{a}{a+b}. \tag{6.5}$$

Recall

Recall, in turn, represents the share of recommended relevant items in the set of all relevant items (cf. [274, p. 12], [84, p. 793], [117, p. 2945]):

$$recall = \frac{a}{a+c}. \tag{6.6}$$

F-Score

In order to express both values with a single measure, the F-score (also known as the F1-measure or F1) is introduced, which represents a harmonic mean of both (cf. [274, p. 12], [84, p. 793],

[117, p. 2945]):

$$Fscore = \frac{2 * precision * recall}{precision + recall}. \tag{6.7}$$

Nevertheless, a separate analysis of the precision and recall values allows for a more differentiated interpretation of the results and as such, the F-score plays a minor role in this work.

6.3.2. Timeliness Deviation – Efficiency of a Recommender System

Timeliness

A major aspect of *Time-Aware Recommender Systems* is to present efficient recommendations based on time features. This means that recommendations should respect the needs of the learners in a timely fashion – supporting the decision process with appropriate recommendations for the given situation. In a time-aware evaluation setting where recommendations should be consumed after they were recommended, the precision value only indicates how many recommended items are relevant to the user.

In a closed-course setting, precision values of items that must be consumed at the point in time of recommendation and those that are relevant at the end of the course would show the same results. While effectiveness can be reported with precision and recall, efficient *TEL* recommendations correspond to the special educational *RS* paradigm. Thereby, the standard evaluation measures do not cover the aspect of the timeliness of the analyzed recommendations. This work introduces a novel, self-devised time-dependent evaluation measure. It borrows established concepts, for instance from the *RMSE* and *MAE*, and represents the "timeliness" measure[106]. The basic idea is that the new timeliness measure indicates how long it takes between the presentation of a recommendation and the time at which the user accesses this item.

6.3.2.1. Mean Absolute Timeliness Deviation

The *Mean Absolute Timeliness Deviation (MATD)*, short *timeliness*, indicates the mean absolute elapsed time for all existing recommendation–consumption value pairs $< tr, tc_i >$ of the presented Top-N list. Thereby, only item category "a" of the confusion matrix (which represents all recommended relevant items) is considered for the analysis of $< tr, tc_i >$:

$$MATD = \frac{\sum_{i=1}^{K} tc_i - tr}{K}, \tag{6.8}$$

where tr represents the point in time of the recommendation of item i presented to a user u and tc_i represents the point of time of the next consumption of item i by the same user u. K is the cardinality of the set of recommended and relevant items. Because of the required *TARS* cross-validation setting, which splits the prediction and test datasets by a time threshold $thresh$,

[106] To the best of the author's knowledge, such a measure (or a similar one for evaluating *Time-Aware Recommender Systems*) has not been published. Thus, it is a new concept, first introduced in this dissertation.

tr must occur before tc_i. All time values must be in the same time unit and refer to the same relative point in time (e.g., as a Unix Timestamp in seconds since January 1, 1970). Information on the time unit must be given alongside the timeliness measure. This allows researchers to better compare different timeliness deviations by converting the given time unit appropriately. In a course setting with course start $tCourseStart$ and course end $tCourseEnd$ points, the following definition applies:

$$tCourseStart < tr \leq thresh < tc_i \leq tCourseEnd. \tag{6.9}$$

If an item has not been consumed after being recommended, it must not be considered for this calculation. The number K reduces in this case to the amount of existing recommendation–consumption value pairs $< tr, tc_i >$. Formally, an item i is only considered if it has been recommended and has been consumed after its recommendation. $r(u, i, tr)$ is a binary function that returns true if user u is given a recommendation for item i at point in time tr. $c(u, i, tc_i)$ is a binary function that returns true if user u consumed item i at time tc_i. The set of recommended and relevant items has a cardinality of K and is defined as:

$$\{i \mid r(u, i, tr) \wedge c(u, j, tc_i) \wedge (tr < tc_i)\}. \tag{6.10}$$

If no recommendation of the Top-N list is relevant, the $MATD$ value should not be considered for further averaging, e.g., for all Top-N lists of all users. However, the share of the non-relevant recommendations within the Top-N item list (that is neglected by the timeliness value) is indicated by the precision value, defined above. This is why a timeliness measure should always be presented in combination with precision and recall.

Similar to other accuracy measurements, single $MATD$ values can be combined, as the mean average, to obtain more general results. It might represent the timeliness of all Top-N recommendations for one user, the timeliness of the recommendations of all users at a specific point in time or even a total timeliness for all Top-N lists of all users over the entire period considered.

6.3.2.2. Cleaned Timeliness Deviation

Practical experiments show that the composition of the dataset can also influence the timeliness value. In the following example, the dataset comprises only two users. User one accessed one item per week – every Thursday. Another user accessed items only at the very beginning of the analyzed period (e.g., during week one) and at the very end (e.g., week 10). Two issues arise:

1. First issue: In a $TARS$ cross-validation with dataset splitting on a per week basis, the chosen week for the splitting threshold affects the timeliness value. If the data are split every Friday, the minimum timeliness for user one would be at least one week until the next

item consumption (from Friday until Thursday). If, in contrast, the chosen splitting day is a Wednesday, the minimum timeliness would be only one day (from Wednesday until Thursday). Thus, the chosen splitting threshold would have a huge impact on the timeliness value.

2. Second issue: When user one (who accessed items on a weekly basis) and user two (who accessed items only at the beginning and the end of the evaluation) are compared, their timeliness values would differ dramatically. That means that the timeliness deviation of user two in week two would be 8 weeks (week 2 until week 10). However, in a real-life setting, when this user was offline the user would not obtain any recommendations before the next use of the system (here in week 10). Thus, also the distribution of consumption data over time affects the evaluation.

Taking both of these extreme cases into account, it makes sense to subtract the time of the next consumption after the analyzed recommendation tc_f from the time of the actual recommendation tr. Item f is the first item that was consumed after tr by the same user ($tr < tc_f$). This period can be formulated per user as:

$$tFirstConsumption = tc_f - tr. \tag{6.11}$$

On the one hand, the $tFirstConsumption$ can be seen as challenge value for the timeliness measure, since it is the lowest possible value a timeliness measure can have ($tFirstConsumption \leq MATD$). Thus, a *Recommender System* aims at forecasting the next consumed item which corresponds to an *MATD* of $tFirstConsumption$. On the other hand, it can also be subtracted from the *MATD* in a so-called cleaned timeliness deviation ($MATD_{cleaned}$). An example of the *MATD*, $tFirstConsumption$ and the cleaned *MATD* is presented in Appendix B.37. This allows for a better comparison of the algorithm results independently from the dataset composition[107]:

$$MATD_{cleaned} = \frac{\sum_{i=1}^{K} tc_i - tr}{K} - (tc_f - tr) = \frac{\sum_{i=1}^{K} tc_i - tc_f}{K}. \tag{6.12}$$

The intuitive approach to this alternative timeliness version is that the time of a recommendation is shifted to be the same as the next item consumption by the user. This circumstance enables researchers to aim at reducing the timeliness measure to zero, which corresponds to the best possible cleaned *Mean Absolute Timeliness Deviation*.

6.3.2.3. Additional Information on the Timeliness Deviation

Remember: In an "increasing time-window" cross-validation setting, the test set size decreases over time by the same number of activities as the training set increases. When further analyzing

[107] The mathematical derivation of the formula is given in Appendix B.38.

the introduced example with 9 weekly splits between the 10 weeks of the course, another issue arises: the timeliness value decreases by definition according to the "increasing time-window" cross-validation. The smaller the test set (after week 1 the test data comprises 9 weeks; after week 9 the test data comprises only 1 week), the smaller the maximum timeliness deviation. This is why the timeliness can additionally be normalized by taking the total duration of the current test set Δt_{test} into account. That builds on the definition of the $MATD_{cleaned}$:

$$MATD_{normalized} = \frac{\sum_{i=1}^{K} tc_i - tc_f}{K * \Delta t_{test}}. \tag{6.13}$$

As a result, the timeliness is given as a percentage of the total available duration. The normalized version lacks information regarding the actual time difference (e.g., the time unit), but can be better expressed in relation to the results of other evaluations. This might help to compare the performance of algorithms in different course settings – for instance, for various course periods.

MATD Examples

Figure 6.2 visualizes a typical example of a course with a period of 10 weeks (100.800 minutes) and 99 users[108]. As the $MATD$ values are averaged, the figure only shows the average timeliness deviation per week for all recommendations within the Top-N lists of all users. On the left side, the absolute timeliness deviation $MATD$, the average duration until the next item consumption $tFirstConsumption$ as well as the resulting $MATD_{cleaned}$ are given in minutes. Thereby, the $MATD$ value cannot be lower than the $tFirstConsumption$ value. The latter builds the challenge line for $MATD$. On the right side, $MATD_{cleaned}$ is additionally normalized by the maximum duration of the test set, that is, given as $MATD_{normalized}$ in percent. As seen from $MATD_{normalized}$, this algorithm performs worst in week 5 and best in week 10 – which is not obvious when analyzing $MATD_{cleaned}$ alone.

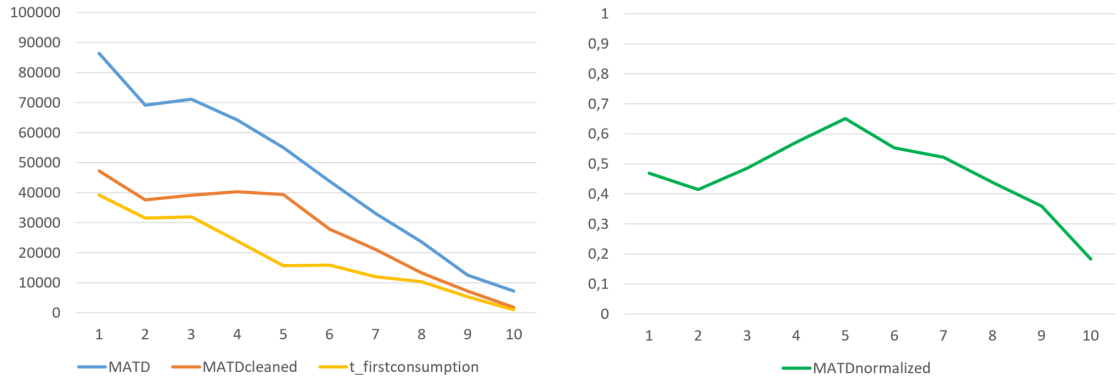

Figure 6.2.: Example for timeliness values over a course period of 10 weeks; left: $MATD$, $MATD_{cleaned}$ and $tFirstConsumption$ given in minutes; right: $MATD_{normalized}$ given in percent of maximum available time per week

[108]The dataset is a simplified version of the AWT dataset.

6.3.2.4. Root Mean Square Timeliness Deviation

The *Root Mean Square Timeliness Deviation (RMSTD)*, its cleaned version, $RMSTD_{cleaned}$ as well as its normalized version, $RMSTD_{normalized}$, are also borrowed from *RMSE* and are defined as:

$$RMSTD = \sqrt{\frac{\sum_{i=1}^{K}(tc_i - tr)^2}{K}}. \tag{6.14}$$

$$RMSTD_{cleaned} = \sqrt{\frac{\sum_{i=1}^{K}(tc_i - tc_f)^2}{K}}. \tag{6.15}$$

$$RMSTD_{normalized} = \sqrt{\frac{\sum_{i=1}^{K}(tc_i - tc_f)^2}{K * \Delta t_{test}}}. \tag{6.16}$$

For both, the *MATD* and the *RMSTD*, the difference between the relevance score of the predicted value and the actual value does not matter – only the fact that an item has been consumed sometime after its recommendation. Thus, the new measure takes the Top-N recommendation list at a particular point in time as the basis. The evaluation setting should be either an "increasing-time window" or a "fixed-time window" cross-validation. *MATD* and *RMSTD*, in turn, represent the mean time errors of recommendations, or in other words, the mean deviation between recommendation and consumption. The calculations are borrowed from *MAE* as well as *RMSE* and show similar strengths: The greater the difference between *MATD* and *RMSTD* (as long as $tc_i - tr \geq 1$ the following applies: $MATD \leq RMSTD$) the more scattered the single deviations. The goal is to reduce the *MATD* and *RMSTD* as much as possible for appropriate time-aware recommendations.

6.4. Applied Evaluation Settings

Based on the definition of the evaluation framework and the underlying measurement values, two main evaluation settings will be applied. These follow similar conditions. For instance, both utilize the *Advanced Web Technologies* dataset. This dataset comprises the biggest number of users and *xAPI* statements. Both settings state numbers on precision, recall and timeliness (cleaned *Mean Absolute Timeliness Deviation*). Their main difference is the definition of items within the test and training set. While Evaluation Setting 1 incorporates the about 1,000 low-level *Learning Objects*, Evaluation Setting 2 comprises just about 100 *Learning Units*. For the sake of comparability, the data of the JavaFX course and of the the Energy-Consultant Training are additionally analyzed in a final evaluation at the end of this dissertation.

Evaluation Setting 1

The main aim is to recommend study material for all items of the course. Thus, it makes sense to recommend items independently of their hierarchy level. A *Learning Object* is restricted to

small learning periods of, at the most, 5 minutes. A *Learning Unit* groups 10 *Learning Objects* on average. In Evaluation Setting 1, the 44,421 "initialized" *xAPI* statements represent the *Learning Object* and *Learning Unit* accesses of the 99 students. The data comprise 1,006 *Learning Objects* and 106 *Learning Units*. That is on average 449 item accesses per user. Appendix B.39 gives the formal definition for this evaluation setting. While this seems to be a reasonable dataset for the evaluation, some algorithms show low performance when processing this amount of data[109].

Evaluation Setting 2

This is why a second evaluation setting is defined for the same course, but with only items from the second hierarchy level. Technically, these are also *Learning Objects* according to the *IMS LOM*, specification. For an easier differentiation, they are called *Learning Units*, and low-level items are called *Learning Objects*. These 106 *Learning Units* comprise the 1,006 *Learning Objects*. However, the "initialized" statements in Evaluation Setting 2 are only considered on the higher level – with in total 8,241 *xAPI* statements. Interactions within a *Learning Unit* are neglected[110]. The formal definition of Evaluation Setting 2 is given in Appendix B.40.

Cross-Validation

For both settings, the course is split, according to the "increasing time-window" validation, into 17 sub-datasets where the time threshold shifts by seven days and is defined per week to be on Mondays at 0 pm. The threshold definition is chosen in order to align the training and test set weeks to calendar weeks. Thereby, with each split, the duration of the training dataset increases by seven days, and the test dataset decreases by the same amount. Appendix B.39 gives information on the data per split in Evaluation Setting 1 and Appendix B.40 gives the same for Evaluation Setting 2.

6.5. Conclusions for Methodology and Evaluation Design

This chapter defined an appropriate evaluation framework for educational *Recommender Systems* that is borrowed from *Time-Aware Recommender Systems*. Thereby, a particular methodology approach has been described and a novel measure, the timeliness, has been introduced.

Scientific Hypothesis /SH5.0/:

An educational *Recommender System* for closed-corpus recommendations may be evaluated by following a specialized evaluation framework in order to produce relevant results. If a traditional evaluation procedure is applied that comprises, e.g., an n-fold cross-validation, its results would not reflect the time-dependent conditions of a closed course. Thus, in order to produce relevant results, a specialized evaluation framework for *Recommender Systems* that aims at predicting course items may utilize a time-dependent cross-validation procedure and may measure the qualitative

[109] For instance, the Slope One algorithm needs over 30 hours for the last iterations of the cross-validation. The complete evaluation over the course with 17 weekly splits took approximately two weeks of calculation time.

[110] The cross-validation of the Slope One algorithm took about 40 hours over the whole course period which makes the algorithm five to ten times faster compared to Evaluation Setting 1.

composition of Top-N lists as well as the timeliness of its recommendations.

The next chapter focuses on the generation of course item recommendations based on *Collaborative Filtering* with time dependency and the evaluation of these recommendations. Thereby, all evaluations follow the defined evaluation framework. Next Chapter

7. Design and Evaluation of Time-dependent Collaborative Filtering

About this Chapter

This and the next chapter introduce the *RS* algorithms that are realized for this dissertation. This chapter starts with a comparison of the results of traditional and time-dependent *Collaborative Filtering* techniques. The idea behind the evaluation procedure is to recommend those learning items in a Top-N manner that the user would consume next, which is a common practice for the evaluation of *Recommender Systems*. The first two approaches are the basic Slope One algorithm as *CF*-baseline[111] and the extended version with incorporated time-weights[112]. Moreover, another *Item-based Collaborative Filtering* approach, the *Time-based Recommender Approach for Lecture Materials (TBRA)*, which is based on time-dependent item similarities for the recommendation of study materials, is adapted and evaluated[113]. Finally, a novel self-devised algorithm is introduced which generates personalized learning paths based on the activities of classmates and the previous interactions of the concerned learner. The next items on the predicted learning path are, therefore, considered as Top-N recommendations. Finally, the four approaches are compared with the help of the time-dependent evaluation framework.

7.1. Traditional Collaborative Filtering

As baseline algorithm, the *Item-based Collaborative Filtering* approach "Slope One" has been chosen, which was introduced in Section 2.2.2.1. It is a simple-to-implement algorithm that produces adequate predictions [175, 278]. Moreover, the algorithm can be easily extended, e.g., through the integration of time weights [142].

7.1.1. Algorithm Design

Slope One in TEL

Verbert et al. tested the standard Slope One algorithm for the recommendation of *Learning*

[111] The traditional Slope One algorithm has been introduced by Lemire et al. [175]. However, Lemire et al. considered this approach only theoretically for *TEL* [174] and, at all, it has only been rarely applied in education contexts (e.g., by Verbert et al. [269]).

[112] The time-weighted Slope One algorithm [142] is applied in this dissertation in the context of *Technology Enhanced Learning* for the first time.

[113] The actual algorithm of presenting similar items [126] has been adapted in this work to fit the recommender's goal and the evaluation approach.

Objects [269, p. 14-15]. Thereby, the Slope One performed similarly to a not further named *User-based Collaborative Filtering* approach and better than another *Item-based Collaborative Filtering* (the Tanimoto-Jaccard Coefficient [261]). Also, Daniel Lemire, who invented Slope One, and his colleagues tested the algorithm for the prediction of *Learning Objects* and, thereby showed its appropriateness for *Technology Enhanced Learning* [174, p. 183-185]. However, due to a lack of appropriate *TEL* data, their evaluation is based on movie datasets and, thus, cannot be generalized or simply transferred to educational settings.

Relevance Score

In a first attempt, the actual algorithm is transferred to the educational domain[114]. Since the *Advanced Web Technologies* dataset does not contain preference rating data, "initialized" *xAPI* statements of the *AWT* course have been incorporated which represent content accesses per learner. Thereby, the relevance scores of the user–item matrix represent the number of times item i has been accessed by user u. In other words: based on the activities of others, the predicted values indicate the number of expected accesses of the particular user per item. Similar approaches build the basis of typical music *Recommender Systems*, such as Spotify [250, 215].

Top-N List

The algorithm predicts the number of item accesses per user for all items, even for those the user has already accessed. Similar to music recommenders, the basic algorithm is, first and foremost, based on the activities of others. All items are ordered according to the predicted relevance score, and the items with the highest values are recommended in Top-N manner.

7.1.2. Evaluation of Traditional Slope One on LOs and LUs

The Slope One algorithm is first evaluated according to Evaluation Setting 1 as introduced in the last chapter (see Appendix B.39 for its formal definition). The algorithm is written in the "R" environment [115] with R Studio[116] which seems to be very slow, especially with increasing matrix density[117]. This is why the results for week 16 are neglected. Different Top-N settings are evaluated where N defines the number of recommended items.

ES1 Precision

In average the precision values are relatively low and range between 0.1 and 0.35 – see Figure 7.1 (blue line on the left). The precision starts with a value of 0.26 and decreases slowly until 0.1 at the course end. However, the precision progress has a peak at week 5 – where the algorithm seems to perform best. By this point, 78 users (out of 99) have accessed at least some items. Thus, the typical cold start phase has ended.

[114] The actual algorithm is based on preference ratings. See Section 2.2.2.1 for an introduction of the Slope One algorithm. Miro Conzelmann initially implemented and started to evaluate this approach as part his *AWT* project titled "Evaluation of Collaborative Filter Systems SlopeOne and Time-weighted SlopeOne Applied To an Online Learning Platform" supervised by Christopher Krauss. An extended version of the evaluation was performed by Christopher Krauss (assisted by the students Dominique Jürgensen and Tolga Karaoglu).

[115] The R Project for Statistical Computing. See: `https://www.r-project.org/` (Accessed: 15.08.2017)

[116] R Studio. See: `https://www.rstudio.com/` (Accessed: 15.08.2017).

[117] While the predictions of week 15 took about 33 hours, the calculation for the last (the 16th) spit aborted after almost four days of calculation at 56%. The performance is shown in Figure B.26. The hardware comprises an Intel Core i5 Quad-Core Processor (4x 3.3 GHz) with 8 GB RAM.

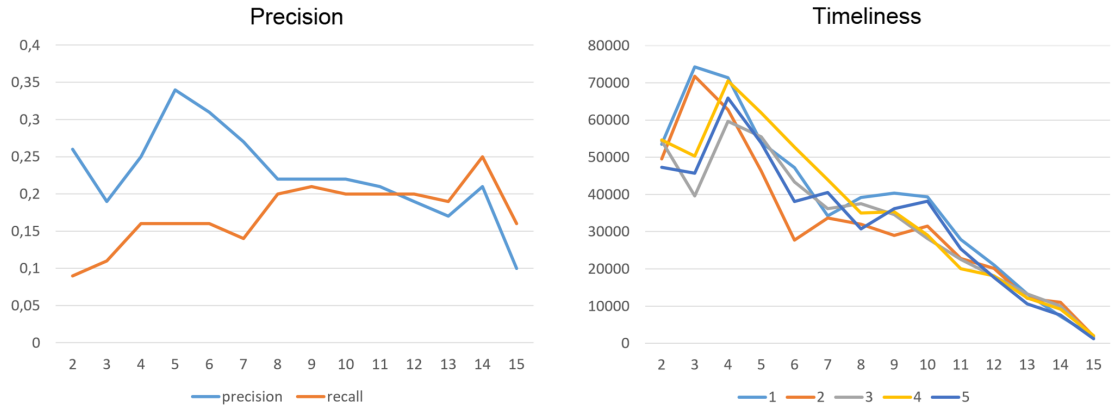

Figure 7.1.: Accuracy of the Slope One algorithm; left: average precision and recall per week of the *AWT* course; right: timeliness values (according to $MATD_{cleaned}$) for different Top-N settings (1 to 5) given in minutes.

The average precision of 0.23 indicates that, in general, only almost a quarter of Top-N recommendations refer to relevant items that have sometimes been consumed after their recommendation. The low precision values might correspond to the fact that the algorithm ignores the information that a user might have previously accessed a recommended item. Thus, items that have been studied before but that were not studied again by the same user after the recommendation reduce the precision dramatically. That effect also explains the reduction at the end of the course, when the test set contains less interactions than previously.

ES1 Recall

The recall values range from 0.09 to 0.25 (Figure 7.1: orange line on the left). In contrast to precision, recall increases slowly over the course period and finally drops off during the last week. The average of 0.17 indicates that about one sixth of relevant items have been recommended in the Top-N list. The drop-off in the last week might be associated with the huge gain in statements during the final learning phase for the exam. Remember, the number of item interactions doubled within the last ten days before the final exam. As a consequence, different access patterns arose that cannot be predicted with similar precision and recall as before.

ES1 Timeliness

The right side of Figure 7.1 shows the timeliness values ($MATD_{cleaned}$) over the course period. The values are given in minutes, where one day corresponds to 1,440 minutes. The five curves refer to different Top-N settings (with a 1 up to 5 recommendations per Top-N list) that all show similar tendencies. In contrast to precision and recall, the timeliness should be as low as possible as this means that the recommendations better fit to the current point in time. The values are between 53 days (Top-5) and 58 days (Top-4) at the beginning and continuously decrease to about five days at the end. The total course period is 112 days (from first access at the *Learning Companion Application* to the final exam), where the highest possible timeliness is restricted to the duration of the test set. The duration of the test set, in turn, is reduced by seven days per week split until it has seven in the last cross-validation split. As can be seen, at week five the timeliness

value is also comparatively low, which corresponds to a high precision value at the same time. Also, the Christmas Holidays (where most users were inactive) can be observed in the timeliness progress, as around week ten the timeliness stagnates and even slowly increases. In average, Top-2 ($MATD_{cleaned} = 32,308 minutes$) and Top-3 ($MATD_{cleaned} = 32,473 minutes$) settings based on Slope One perform similarly well with different local valleys (Top-3 in week 3 and Top-2 in week 6). Top-1 recommendations, in contrast, perform worse ($MATD_{cleaned} = 37,486 minutes$), as recommendations have in this case been consumed about 3 to 4 days later in average.

7.1.3. Evaluation of Extended Slope One on LUs

ES2 with Learning Units

In a second attempt, the effect of incorporating only *Learning Units* instead of *Learning Objects* together with *Learning Units* is analyzed. It corresponds to Evaluation Setting 2 (see Appendix B.40). Surprisingly, this has a significant effect on the precision – as shown in Figure 7.2. While this approach shows a similar overall trend including the cold start phase, a slow decrease of the precision over time and a huge drop-off at the end, the average precision improved by over 23% to an average of 0.459. Moreover, also the timeliness improved but on a smaller scale by 12.2% on average. This is because the user–item matrix for *Learning Units* is denser than the original user–item matrix. On a higher level, the users provide more feedback per item on average, because in *LCA* users need to first access a high-level item and then decide on a low-level item. Consequently, there are more interactions per user–item tuple in the database which leads to more precise recommendations. This comes at the cost of less offered items – each on a higher level and thus a broader topical scope: One *Learning Unit* contains about 10 *Learning Objects* on average. However, 106 unique *Learning Units* per course still require the help of a *Recommender System* for an efficient item selection.

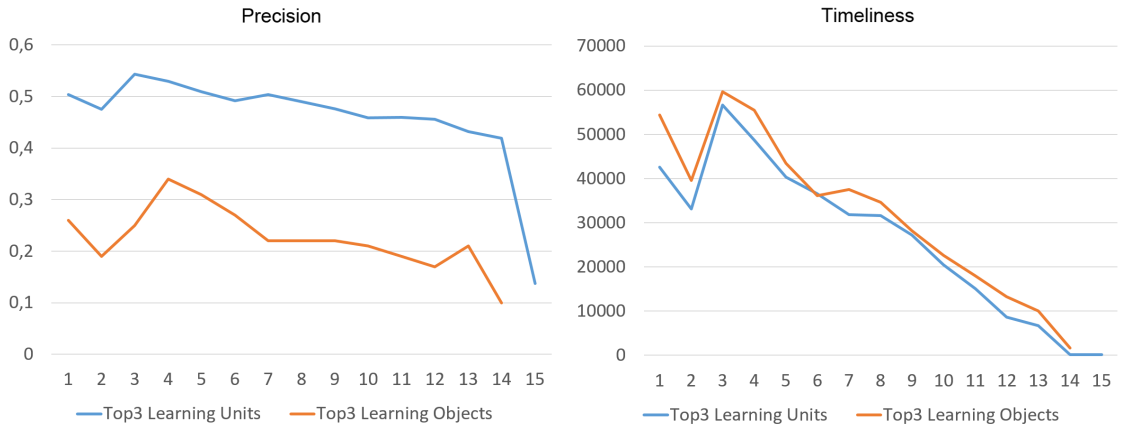

Figure 7.2.: Accuracy of the Slope One algorithm on *Learning Units* (blue) and *Learning Objects* (orange); left: average precision per week of the *AWT* course; right: timeliness values (according to $MATD_{cleaned}$) given in minutes.

Extended Educational Slope One

Another improvement comes by considering the specialized *TEL* paradigm: thus, the Slope One algorithm should represent a learning need instead of a simple activity forecast. The same approach as introduced before is applied to predict the number of user u's accesses to item i based on similar users – this is called $predictedAccesses_{u,i}$. The deviation between this predicted value and the real number of item accesses $realAccesses_{u,i}$ for the same user and item represents the new relevance score:

$$relevanceScore_{u,i} = predictedAccesses_{u,i} - realAccesses_{u,i}. \tag{7.1}$$

The idea behind this approach is to reduce the prediction value, which reflects the accesses (implicitly the "interests") of others, by the number of the user's actual accesses. In general, a prediction of an already known rating or access number would not make much sense for e-commerce or entertainment services, as it is previously known and does not need to be predicted. However, due to the specialized paradigm, the user–item matrix of a closed-course *Recommender System* is expected to be much denser. For the prediction step of this algorithm, the real number of accesses of a particular user on an item is treated as unknown. This approach helps to compare the activities of others with the user's activities. Given the example of a Slope One item score prediction of 5 for item i and a predicted score of 7 for item j. Thus, based on the behavior of others, item i should have been accessed 5 times and item j 7 times by now. The traditional Slope One algorithm would prefer item j since its relevance score is higher – implicitly other users commonly accessed item j more often than item i. Nevertheless, when the user has accessed item i already 2 times and item j has been accessed 5 times, the corresponding relevance score is 3 for item i and 2 for item j. This deviation value represents the new relevance score: how much a user must study/access a *Learning Unit* to keep up with the classmates.

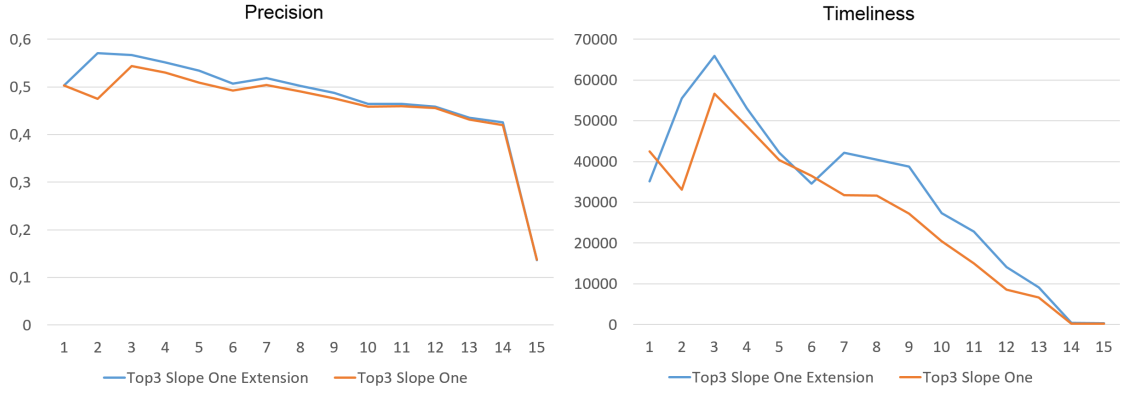

Figure 7.3.: Accuracy of the extended Slope One algorithm (blue) and the traditional Slope One (orange); left: average precision per week of the *AWT* course; right: timeliness values (according to $MATD_{cleaned}$) given in minutes.

Evaluation of Extension

This approach has been applied to the *Learning Units* of Evaluation Setting 2. The results are

shown in Figure 7.3. As can been see, the extended approach has a positive effect on the precision which leads to an improvement of on average 1.6% (0.475 compared to before reached 0.459%). Especially in the beginning of the course, the approach shows its strengths. However, the positive effect lessens over the period to the course[118]. A possible reason for this lies in the dataset: As discussed in Section 5.3, successful students (of the completing cluster) start earlier to study the topics compared to the other students. In the beginning, the extended Slope One algorithm processes more interaction data regarding learners that were frequently active over the entire course period. The positive effect lessens the more infrequently learners use the *LCA*. Moreover, no user accessed all available items and most users consumed only a few of them. Only some students repeated a before studied item which means that the average number of *realAccesses* is very low and a differentiation between the item accesses of users is tough. Finally, the timeliness in Figure 7.3 indicates an overall negative effect with higher values of *MATD* when comparing the Slope One extension with the original algorithm.

7.2. Time-based Slope One Algorithm

The Slope One improvement of Jiang and Lu [142] considers the time of the given item feedback. Thereby, the Time-based Slope One approach borrows the concept of Li et al. [177] who introduced different neighbor weights to the original approach.

7.2.1. Algorithm Design

The time-weighted Slope One algorithm is based on the original version presented by Lemire and Maclachlan [175] (introduced in Section 2.2.2.1). It is divided into two steps: (1) calculation of the score deviation of other users from the item of interest and (2) prediction of a new score based on the previously determined deviations.

Step One

Jiang and Lu extended the first step and therefore incorporated time weights $tw_{u,i1,i2}$ that correspond to the timespan between the accesses of item $i1$ and $i2$ (e.g., the point in time when the user accessed or rated the two items) [142, p. 2296]. According to Jiang and Lu the deviation of step one is adapted as:

$$devTW(i_1, i_2) = \frac{\sum_{u \in U_{i_1 i_2}} (r_{u,i_1} - r_{u,i_2}) * tw_{u,i1,i2}}{|U_{i_1 i_2}|}, \tag{7.2}$$

where r_{u,i_1} is the item's rating user u gave. $U_{i_1 i_2}$ is the set of users who rated both items and $|U_{i_1 i_2}|$ is its cardinality. A new function, the time weight $tw_{u,i1,i2}$, has been introduced which transfers the timely deviation of the two times of item accesses $t_{u,i1}$ and $t_{u,i2}$ of user u into a

[118] The incorporation of the $realAccesses_{u,i}$ value can have a huge effect on the Slope One algorithm – even though a negative when applying the reverse formula accidentally (see Appendix B.43).

weight. The weight is high when both points in time are similar (up to 1 if they are equal) and low, the more time lies between the two item accesses.

$$tw_{u,i1,i2} = \frac{1}{1 + \alpha * \Delta t_{rel}}. \tag{7.3}$$

Δt_{rel} is defined as the relative deviation between item accesses $t_{u,i1}$ and $t_{u,i2}$. Relative means that the deviation is normalized by the highest possible time value tr:

$$\Delta t_{rel} = \frac{|t_{u,i1} - t_{u,i2}|}{tr}. \tag{7.4}$$

In this implementation, tr is the point in time of the recommendation and all three time values (tr, $t_{u,i1}$ and $t_{u,i2}$) are given in milliseconds since course start. If an item has been accessed multiple times by the same user, only the last access is considered for $t_{u,i1}$ or $t_{u,i2}$.

The parameter α adjusts the effect of time decay. If $\alpha = 0$, the algorithm works as known from Lemire and Maclachan [175]. The higher α the faster older ratings are forgotten. Figure 7.4 visualizes the effect of different settings of α.

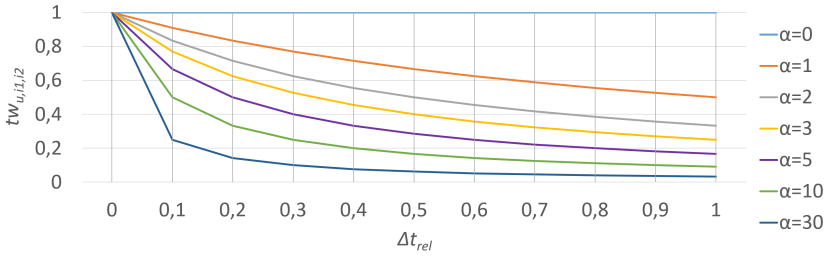

Figure 7.4.: Effect of different α settings: the x-axis displays the relative timespan between the two analyzed items and the y-axis shows the corresponding time weights.

Step two remains the same except for incorporating the adapted time-dependent deviation $devTW$ instead of the standard Slope One item deviation. The algorithm predicts the relevance score $preTW(u, j)$ for user u on item j: Step Two

$$preTW(u, j) = \frac{\sum_{i \in Ij}(r_{u,i} - devTW(i, j))}{|Ij|}. \tag{7.5}$$

Ij is the set of all relevant items to be compared with item j and $|Ij|$ is its cardinality.

7.2.2. Evaluation

The time-weighted Slope-One algorithm is evaluated according to Evaluation Setting 2[119]. Figure 7.5 visualizes the precision and timeliness results.

[119]The analysis of the LO activities in Evaluation Setting 1 can be seen in Appendix B.44.

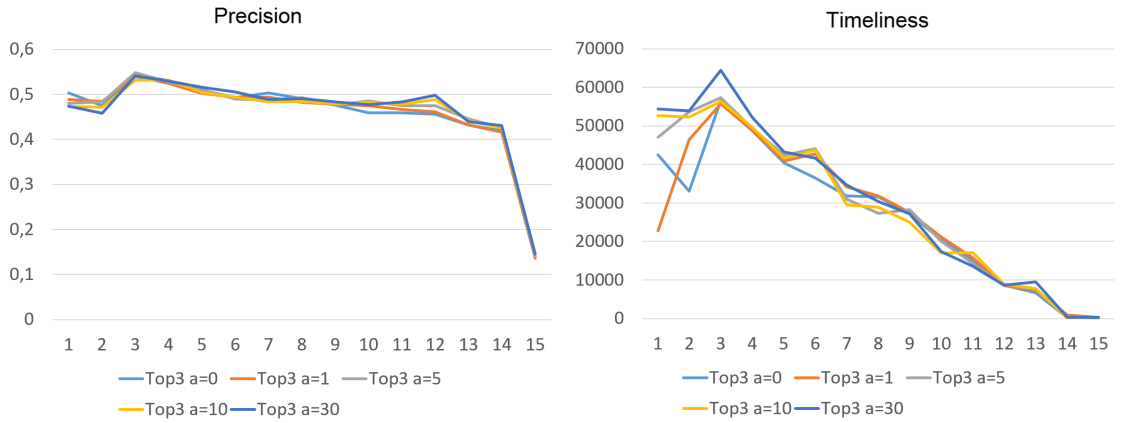

Figure 7.5.: Comparison of Time-based Slope One algorithm ($\alpha > 0$) compared with the regular algorithm ($\alpha = 0$); left: average precision per week; right: timeliness values (according to $MATD_{cleaned}$) given in minutes.

As can be seen, the precision has improved by only 0.56% at most ($\alpha = 30$) and the timeliness value is even downgraded by up to 11.5%. Additionally, the two extensions have been combined: the algorithm extension that deviates the real number of item accesses and the extension with time-weights. The combination brings a total improvement to the precision of 2.6% compared to the original Slope One on *Learning Units*. While the performance is similarly slow for both approaches[120], it must be concluded that the extended version and the time-weighted version do not bring the expected improvements to the algorithm.

7.3. A Time-based Recommender Approach for Lecture Materials

The *Time-based Recommender Approach for Lecture Materials (TBRA)* is the only published *Time-Aware Recommender System* for course contents [126] – as presented in Section 3.5.2.4. It is based on similar considerations to those of the extended Slope One implementation since it incorporates the number of accesses per item as well as information on access times. However, it requires an additional pre-processing step for the generation of an item similarity matrix.

7.3.1. Algorithm Design

Hermann used this approach to present related items to a previously accessed lecture material [126]. The published approach is extended in this work to recommend items in a Top-N manner and consists of the following five steps:

[120] Appendix B.45 presents a performance comparison of both Slope One settings.

1. Based on previous item accesses of all learners, an overall similarity matrix is computed. This relates all course items with each other and comprises the activity data of all learners in the course. Each item–item tuple $< i, j >$ contains a similarity score $S_{i,j}$ in the range [0,1]. The similarity score is based on the timespan between the access times of both items per user. If a user accessed an item several times, only the minimum timespan is considered for $< i, j >$. This timespan is then averaged for all users who accessed both items, where 1 would mean both items are in average accessed at exactly the same point in time by the same user and 0 means their access times are totally diverse. An example of the item–item matrix might look like this:

Table 7.1.: Example of the TBRA similarity matrix averaged over all users

	i_1	i_2	i_3
i_1	1	0.3	0.1
i_2	0.3	1	0.7
i_3	0.1	0.7	1

This matrix indicates which items are frequently accessed in conjunction, e.g., within a course. In the example, $i2$ and $i3$ are more often accessed within a short time compared to, e.g., $i1$ and $i3$. This step corresponds to the basic approach of Hermann [126][121]. In comparison to Slope One, it can be computed offline – as soon as the cold start phase is over and there is a certain amount of interaction data.

2. In parallel, a user–item matrix is generated ($A_{u,i}$), where each user–item tuple $< u, i >$ represents the number of item i accesses of user u. An example is given below:

Table 7.2.: Example of the TBRA user-item-matrix for accesses

	i_1	i_2	i_3
u_1	5	1	0
u_2	2	0	0
u_3	0	0	1

3. In a third step, the item list is cascaded. Instead of recommending similar items to the actually presented one (as Hermann does [126]), only items that show the lowest access rates are considered as recommendations. Thereby, the lowest access value per user in the user–item matrix serves as the threshold for the Top-N recommendations. In other words,

[121] The students Kumar Awanish, Gaurav Vashisth and Rudresha Gulaganjihalli Parameshappa implemented and evaluated a version of this algorithm for their *AWT* project "Realization and Evaluation of a Time-Aware Recommender System for Lecture Materials" supervised by Christopher Krauss. After the project submission, Christopher Krauss extended the approach and evaluation to fit the needs of the Top-N evaluation setting introduced here.

as long as there is one item that has never been consumed by the current user, the personal access threshold remains under the value of one and all items that have been consumed at least once are not recommended.

4. All items with the lowest access value per user obtain a relevance score that refers to the similarity (given in the similarity matrix) of the most frequently consumed items by the user:

$$TBRA_{relevance}(u,i) = \frac{\sum_{j=0}^{n}(S_{i,j} * A_{u,j})}{n}, \tag{7.6}$$

where n is the number of considered items. Items above the threshold are given a relevance of 0. The resulting user–item matrix with relevance scores for the example is presented below:

Table 7.3.: Example of the TBRA user-item-matrix with relevance scores

	i_1	i_2	i_3
u_1	0	0	$\frac{0.1*5+0.7*1+1*0}{3} = 0.4$
u_2	0	$\frac{0.3*2+1*0+0.7*0}{3} = 0.2$	$\frac{0.1*2+0.7*0+1*0}{3} = 0.07$
u_3	$\frac{1*0+0.3*0+0.1*1}{3} = 0.03$	$\frac{0.3*0+1*0+0.7*1}{3} = 0.23$	0

5. Finally, the items are ordered per user according to their relevance score $TBRA_{relevance}$. Only the Top-N items are recommended. When presenting just the Top-1 item in the example, user u_1 is given a recommendation for item i_3, user u_2 for item i_2 and, consequently, user u_3 should study item i_2 as well.

7.3.2. Evaluation

For reasons of comparability, the evaluation setting follows the rules defined in Setting 2 with just *Learning Units*. Figure 7.6 presents the results for precision, recall, and timeliness for Top-3 recommendations which performed best with the Slope One algorithm. Moreover, the Top-10 and Top-30 results are shown as well, since a higher number of recommendations appears to have a positive effect on the timeliness.

Precision

The three lines at the top of the left side of Figure 7.6 show the precision of *TBRA*. The Top-3 setting shows the highest coverage of relevant items in the Top-N list. At the beginning of the course, the precision lies between 0.57 and 0.69. However, these numbers fall continuously over the course period with some local valleys (e.g., in week 6 and week 12 for Top-3) and a huge drop-off at the end of the course. Within the final week, the precision is reduced for all settings to about 0.25. The average precision shows a coverage of 0.51 – which means that half of the recommendations within the Top-3 item list are not relevant. Two major reasons have been identified:

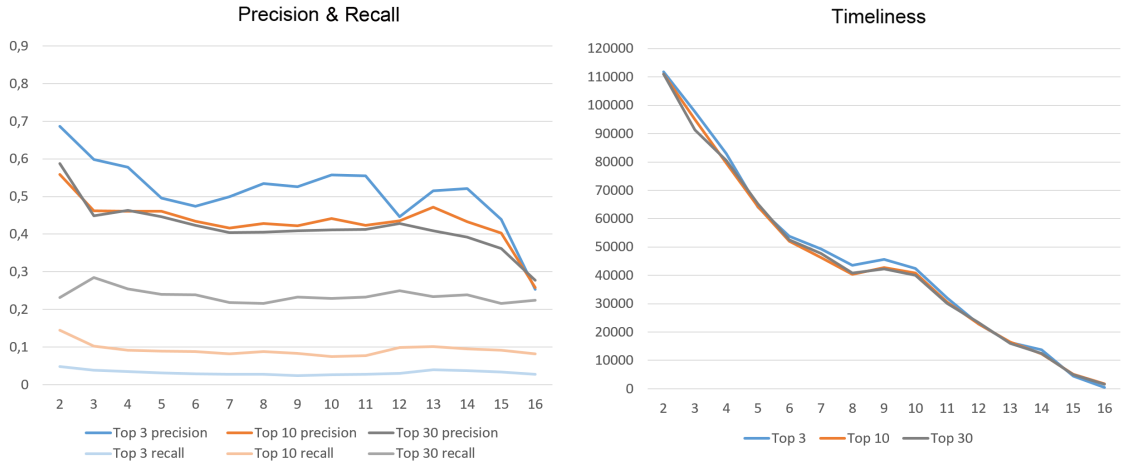

Figure 7.6.: Evaluation results for the *TBRA* for Top-3, Top-10 and Top-30 recommendations considering all users; left: average precision and recall; right: Timeliness values for averaged $MATD_{cleaned}$ given in minutes.

1. Learners who did not provide any feedback cannot obtain any recommendations based on their behavior – this corresponds to the well-known cold start problem. According to the algorithm, every item is given a relevance score of 0, and the Top-N list comprises random elements.
2. Moreover, learners who accessed all items do not necessarily start over and study items right from the beginning nor in the same order as before. However, the algorithm recommends items in the same order even for a second learning iteration through all items, because all items that have been accessed least are considered as recommendation. This circumstance describes the precision drop-off at the course end.

Considering these two aspects, a second evaluation is performed in a adapted version of Evaluation Setting 2. Only those learners are considered who accessed at least one item and who did not access them all[122]. Figure 7.7 shows the results of this second attempt. As one can see, the analysis shows a very high degree of precision – for the Top-3 setting in average 0.72. That means, almost three quarter of the items recommended in the Top-3 list are relevant to the user – an improvement of more than 20%. The Top-3 setting shows three local peaks (in week 4, week 10 and week 14) and ranges between 0.57 and 0.83 and the Top-10 as well as the Top-30 setting also show some improvement of about 12% to 14% and a more stable trend compared to the Top-3 setting. Their average precision values range from 0.57 for Top-10 to 0.54 for Top-30 recommendations. It can be concluded that the precision of the algorithm depends heavily on the access patterns of particular users. Since the users accessed 38 unique items on average, the precision is affected by

[122]The number of considered users per week can be seen in Table B.2 in Appendix B.40 (users of the training and test set). For example, instead of 99 learners, only 24 learners are considered for split one, 50 for split two, 63 for split three and so on. First users have accessed all items at the beginning of the last two weeks. Thus, only the last two splits are affected by omitting learners who accessed all items.

those users who accessed fewer items than recommended in the Top-N list – 22% of users in AWT accessed less than 10 *Learning Units* and even 57% of users accessed less than 30 LUs.

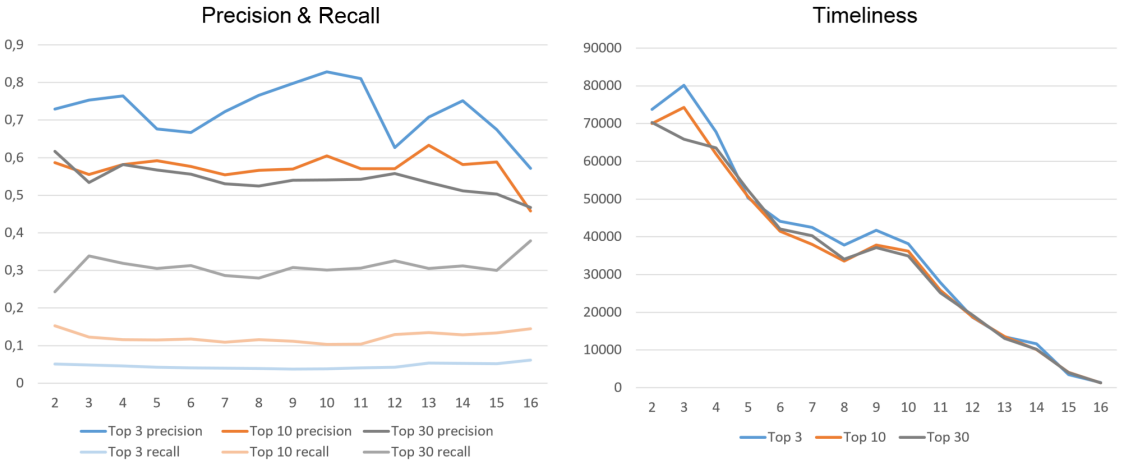

Figure 7.7.: Evaluation results for the optimized *TBRA* for Top-3, Top-10 and Top-30 recommendations considering only users who accessed at least one item and not all items; left: average precision and recall; right: Timeliness values for averaged $MATD_{cleaned}$ given in minutes.

Recall

In contrast to the high precision values, the recall values are comparatively low for both settings. Of course, the fewer the number of items recommended, the smaller the recall value. Especially in the Top-3 environment, only 5% of relevant items are part of the Top-N list. Top-30, in turn, shows an average recall of up to 0.30 in the optimized setting. However, the optimization has only a small positive impact on the recall value.

Timeliness

The right sides of Figures 7.6 and 7.7 show the timeliness measurements. Besides some differences in the beginning until the Christmas Holidays (up to week 10), the timeliness measures are almost similar in both evaluations for the three Top-N settings.

When comparing the attempt with all users and the attempt with well-selected users (only those learners who accessed at least one item and who did not access them all), another advantage is uncovered. Especially at the beginning of the course, the optimized version is more accurate in time as the recommendations have been accessed 28 days, on average, before those of the first attempt. However, after the first half of the course, this effect decreases and the timeliness of both versions converge. The timeliness of the Top-3 setting with well-selected learners improves in average by six days over the entire period.

Top-30 performs best ($MATD_{cleaned} = 34,235$), followed by Top-10 with an almost equal average Timeliness value ($MATD_{cleaned} = 34,515$). Thereby, a higher number of recommended items seems to have a positive effect during the cold start phase, which can be observed until week 4. However, also the Top-3 setting ($MATD_{cleaned} = 36,852$) shows a good result that is only one up to two days behind the other settings.

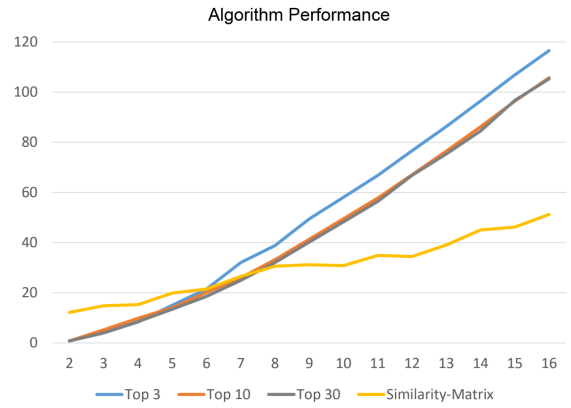

Figure 7.8.: Computation times (y-axis) for *TBRA* for Top-3, Top-10 and Top-30 recommendations over all users and the mean computation time for the generation of the similarity matrix given in seconds; x-axis presents the course period in weeks.

Performance

The computation time in Figure 7.8 is given in seconds. There is no huge effect of the different Top-N settings, because a prediction for every user–item tuple has to be computed independently from the number of recommended items. Taking into account that the similarity matrix can be computed offline and at regular intervals, the aim would be to perform the recommendation step online. However, the generation of Top-N recommendations per user is in the worst case slower than one second (about 120 seconds for 99 users) and with a longer course duration or more items that need to be studied, the calculation duration increases. Thus, the prediction step should be performed offline and in regular intervals, as well.

Conclusions for TBRA

The evaluation of the *TBRA* algorithm indicates that the information on previously learned items has a crucial impact on the appropriateness of its recommendations – as shown by the comparison of all and only well-selected learners. Obviously, learners prefer to access new items instead of repeating previously accessed ones. When cutting off the users that did not access any item (which is only a theoretical setting), the precision improves at the cost of worse timeliness. Taking the high precision value into account, the Top-3 settings performs best in this evaluation which is similar to the findings of the Slope One evaluation.

7.4. Recommendations through Learning Path Predictions

As the point in time and the information on previously studied items have a significant impact on the precision and timeliness of recommendations, an additional recommender approach is tested which focuses more on item sequences. Given the number of about 100 *Learning Units* within the *Advanced Web Technologies* course, there are about 10^{157} possible combinations of item sequences ($100! \approx 9 * 10^{157}$) – not considering the about 1,000 *Learning Objects* on a lower hierarchy level. The learner, in turn, only needs to select one appropriate series, that is called learning path, to

consume all given items. The approach to predicting these personalized learning paths consists of two separate steps:

1. The first step handles the definition of the knowledge graph that represents possible routes through the items. A hybrid combination of teacher-based, constraint-based and user-interaction-based approaches are utilized for building the knowledge graph[123]. The direct transitions (edges) from one item (node) to another comprises a probability that describes the percentage of past transitions. Thereby, each historical path of learners is analyzed to create a knowledge graph and the probabilities. In other words: the more often users studied items in a particular order, the higher is the probability that this part of the path will be recommended to other learners in the future.
2. The second step predicts an individual learning path for each learner in the given situation. Thereby, the past item accesses are considered for the requesting user to predict a personalized subpath. This future path starts at the last consumed item node and avoids other, previously seen, items. The idea is that the algorithm searches for the shortest (and most efficient) path through all left learning items in the before built knowledge graph.

In general, this approach is different to classical *Collaborative Filtering* approaches where items might be recommended independently of the current learning context and direct dependencies on other topics. However, it incorporates collaborative data and, thus, is treated as a *CF* approach in this work. The result is a set of individual paths that lead through all offered items exactly once. Repeating or skipping items is not possible when consequently following the recommended route. However, an extension of step 2 might neglect items that are marked as "known" by the learner – which is not implemented for this evaluation.

7.4.1. Algorithm Design

The algorithm is inspired by routing plan algorithms for public transportation where trains follow particular schedules [304][124]. The actual route of a passenger, however, is similar to the learner's individual learning path in that it needs to follow some constraints (e.g., the lecture schedule) but allows for decisions at multiple points in time in respect of alternative routes.

Dependency Graph

The actual algorithm splits step 1 into two parts. At first, the *AWT Learning Units* are transferred into a graph database as nodes. Let I be a set of items belonging to the course C.

Prerequisites, given as attributes in the *IMS Learning Resource Meta-data Specification* [1], are considered for a primary dependency graph. Prerequisites for the *AWT Learning Units* are

[123] A graph can be built based on different strategies as introduced in Section 3.6.

[124] This work is part of the Master Thesis of Andreas Salzmann with the title "Leveraging Time-Dependent Multi-Modal Routing Plan Algorithms to Create Personalized Learning Paths for Technology Enhanced Learning". The idea and the concept of learning path generations as well as the methodology were defined by the day-to-day-supervisor of the thesis, Christopher Krauss. However, the implementation of the algorithm, as well as the execution of the evaluation steps, were done by Andreas Salzmann.

manually defined and can be seen as constraints of a path that must not be violated – for instance the *Learning Unit* "Concepts of Recommender Systems" is a prerequisite for the *Learning Unit* "Technical Issues in Recommender Systems". The result is a dependency graph representing all allowed learning transitions. See Figure 7.9 for an example of the dependency graph with six items $(LO1, ..., LO6)$ where $LO2$ is a prerequisite for $LO3$ ($LO3 \implies LO2$ [125]), $LO3$ is a prerequisite for $LO4$ ($LO4 \implies LO3$) and $LO5$ a prerequisite for $LO6$ ($LO6 \implies LO5$).

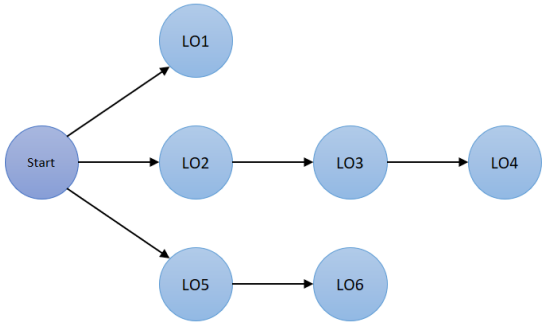

Figure 7.9.: Example of a dependency graph with six LOs and three defined prerequisites ($LO4 \implies LO3$, $LO3 \implies LO2$ and $LO6 \implies LO5$).

Knowledge Graph

Secondly, based on the defined prerequisite constraints, all allowed combinations of items in I are listed. The subset s of items in I represents one possible learning state with all so-far consumed items. S is the set of all possible states within the course:

$$S = \{s | s \subseteq C\}. \tag{7.7}$$

Each transition from one state to another is represented by a directed edge $e_{s_1,s_2} \in E$ except where the transition violates the prerequisite definition. An edge connects two states s_1 and s_2 where one item is added to s_1 which results in state s_2. E is the set of all edges.

The directed transition edges are extended by an attribute that indicates the number of other learners who used the same transition in the past. More precisely, it represents a ratio of historic transitions $e_{transitions}$ between exactly the two states (s_1 and s_2) and the number of all historic transitions $s_{transitions}$ leaving s_1. The result of the probability function $p(e)$ determines the transition probability per edge e of historic movements of all learners and is given in percent:

$$p(e) = \frac{e_{transitions}}{s_{transitions}}. \tag{7.8}$$

As an intermediate result, a knowledge graph K is generated that represents all possible states and transitions of the dependency graph between a start state $B = \emptyset$, which does not contain any

[125] Could be interpreted as "$LO3$ implies / requires the knowledge of $LO2$".

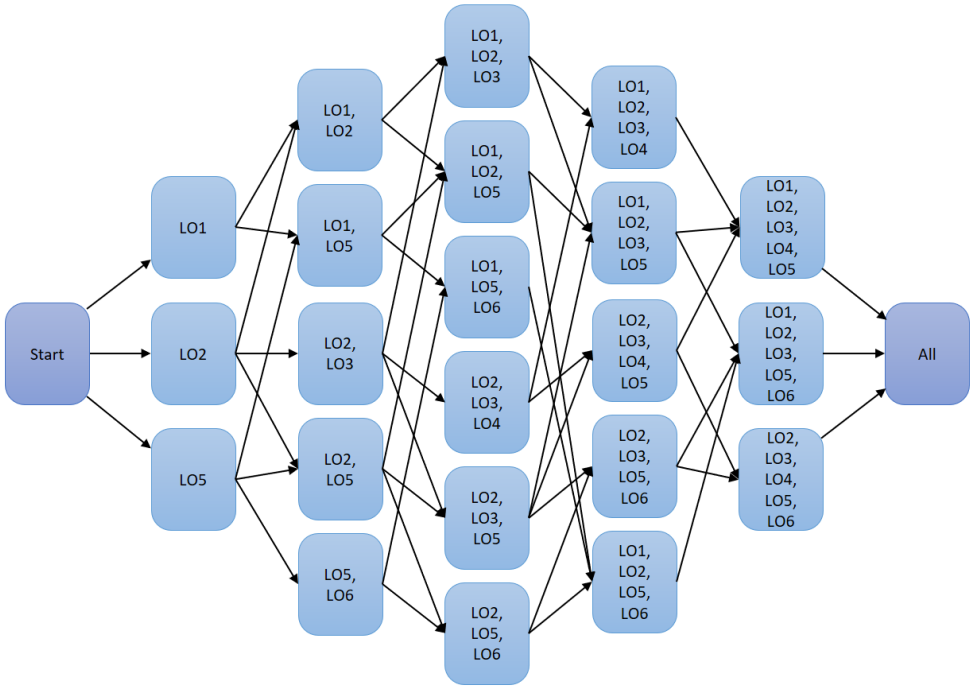

Figure 7.10.: Example of a knowledge graph with the same six example *LOs* of the dependency graph example.

item, and a target state $T = C$, which contains all items of the course.

$$K = (B, T, S, E, p). \tag{7.9}$$

Figure 7.10 visualizes the example knowledge graph with the six items of the dependency graph example. This graph is computed offline and stored in a database to efficiently calculate individual paths in the future.

Path Generation

When now a learner requests a path recommendation, all of the so far accessed items of that user are considered. The learner's accessed items correspond to a state in the knowledge graph K reflecting a new starting point $B_{u,t}$ for user u at time t. When the user, for instance, accessed $LO3$, $LO2$ and $LO4$, the state $\{LO2, LO3, LO4\}$ is the starting point for a shortest-path algorithm. If the user violated the prerequisite definition and studied another item first, the state that shows the highest agreement of known items is considered as the start state.

The shortest-path algorithm identifies an ideal route $r \in R$ from the start to the target point T in the knowledge graph (marked with "All" in Figure 7.10). The target point T reflects all studied items in the course and thus the course goal. R is the set of all possible routes between B and T. Thereby, the results of the probability function $p(e)$ per taken edge e on the route are summed. E_r is the number of edges of the route r. The path with the highest total probability $p_{total}(r)$

should be preferred.

$$p_{total}(r) = \sum_{e=0}^{E_r} p(e). \tag{7.10}$$

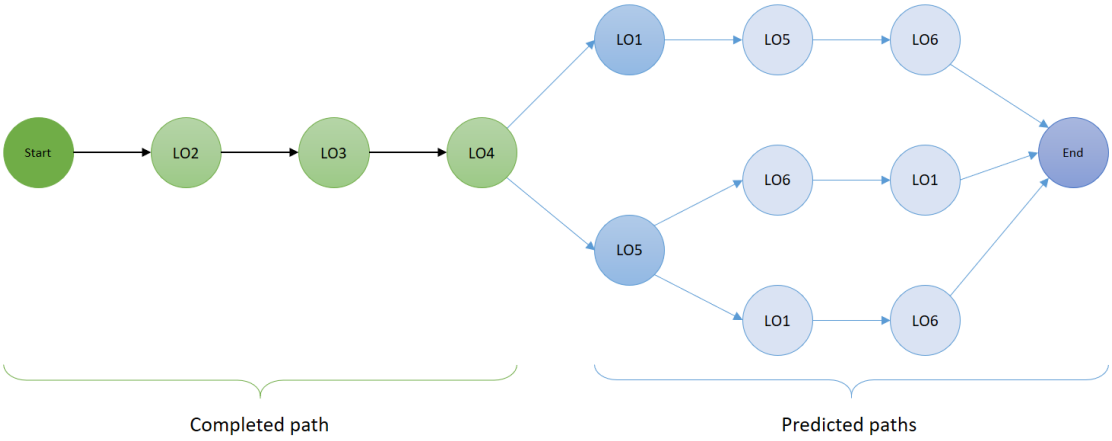

Figure 7.11.: Example of a path presented to the learner when LO2, LO3 and LO4 have been consumed.

The routing plan algorithm of Zografos and Androutsopoulos [304] was used in the evaluation. It analyzes backwards the transitions from T to $B_{u,t}$ by considering the probability $p(e)$ per edge. Moreover, as the algorithm is designed for public transportation means, it allows incorporating waiting times, when changing the transportation. Transferred to the learning domain, this feature is used to penalize switches of the higher-level topics as the learner might better focus when working on similar topics at a time and encourage switches first when all low-level items belonging to one high-level item are processed. The result is a list of alternative routes with a number of branches b per node. Only those b branches are considered that show the highest transition probability $p(e)$ from one edge to another[126].

Experts can adjust the total number of considered and presented branches. For the evaluation, different numbers of branches b and also of presented Top-N items are considered. The implementation is realized as Java server with the graph database neo4j[127].

7.4.2. Evaluation

In order to be compliant with the defined evaluation settings, the knowledge graph is generated only by processing the data of the training set. Thus, the transition probabilities have a minor

[126] Andreas Salzmann used in his Master Thesis two additional types of sequence alignment algorithms to present branches, borrowed from bioinformatics: the Needleman-Wunsch algorithm for global alignments [201] and the Smith-Waterman algorithm for local alignments [248]. However, these algorithms are not applied for this evaluation and, thus, are outside of the scope for this work.

[127] neo4j. See `https://neo4j.com/` (Accessed: 22.08.2017).

effect at the beginning of the course where recommendations rely, first and foremost, on the defined prerequisite dependencies. However, the more activity data are collected, the higher is the effect of the transition probabilities. Indeed, in a real-course setting, all available activity data (especially the data of historic courses) would be incorporated.

Precision, Recall & Timeliness

Precision, recall, timeliness (given in minutes) as well as performance measures are determined according to Evaluation Setting 2. Besides the split per week, which lead to 15 different time-dependent results, different sizes of branches per node are considered, which are given as b ($b = 1, 2, 3, 5, 10$ and 15). Moreover, the Top-N lists are also analyzed with different sizes – $3, 5, 10$ and 15.

Evaluated Top-N Lists

Since the number of branches b and recommendations N are not necessarily equal, the Top-N list consists of the items with the lowest distance (number of edges) to the last accessed item (state $B_{u,t}$). When deciding for items with the same number of edges to the last accessed item, those with the highest transition probability are preferred. This is determined iteratively, starting with direct connected items. Given the example of $b = 3$ and $N = 4$, the 3 direct connected items ($numberOfEdges = 1$) are set on the Top-N list in the first iteration. Afterward, all 9 items ($b^{numberOfEdges}$) that are connected via 2 edges to the start item are considered as the missing Top-N items. Thereby, the algorithm prefers the items with the highest transition probability. This is repeated until the Top-N list consists of N items where every item of the Top-N list must be unique. For the defined evaluation setting and in order to produce comparable results with the other approaches, the connections between the items (the actual learning path) is not of interest for the measurements (just for the selection process). For the evaluation, only the fact is evaluated that an item is part of the Top-N list.

Finally, for reasons of comparability, a Top-N list is analyzed consisting of all relevant items which means the recall value is 1. Remember, an item is relevant if it has been accessed by the user after the point in time of the recommendation. Thus, the validation was performed in 450 iterations (15 week splits * 5 Top-N settings * 6 branch settings)[128]. Due to the vast amount of experiments, the first figures show the precision and recall of the main settings averaged over the whole course period. This is done to better compare the different settings. At the end, the best settings are presented, as already known, over the period of the course.

Precision & Recall

Figure 7.12 shows the precision and recall of the main settings. In general, the fewer the number of elements in the Top-N list, the more precise are the recommendations (Top-3 performs best when $b > 2$) and the lower the recall (where Top-3 performs worst). Over half of the recommended items are relevant (precision of up to 0.58) and almost 25% of relevant items have been recommended as the recall value indicates. Except for the setting with 15 branches ($b = 15$), the analysis shows a

[128] Note: as this approach relies on past feedback of other learners and item dependencies, it allows for the recommendation of learning items right from the beginning and does not require any initial user feedback. That is why first recommendations are presented before week 1.

high dependency of precision and recall on the amount of Top-N items.

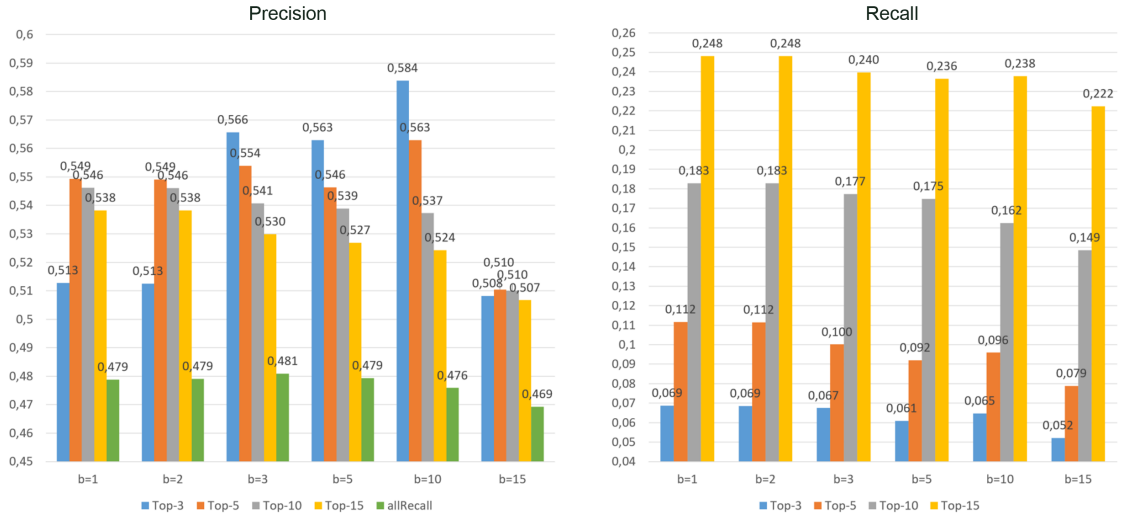

Figure 7.12.: Precision (left) and recall (right) for different settings of the Top-N path recommendations.

The number of branches, in contrast, does not have such a massive impact on the precision measure as the Top-N elements have. Nevertheless, it seems clear that $b = 10$ is the best setting – and $b = 1$ and $b = 2$ perform almost identically low. For recall, the b-settings have practically no effect. As expected, the number of 15 recommendations in the Top-N list performs best.

The highest F-scores (defined as the harmonic mean of precision and recall) are determined for $b = 2$ and Top-15 (F1 of 12.319%), $b = 1$ and Top-10 (F1 of 12.318%) and $b = 3$ and Top-15 (F1 of 11.500%) due to the significant impact of the recall value.

Timeliness

Keep in mind: in contrast to the prediction measures, the cleaned timeliness should be as low as possible. Figure 7.13 gives an overview of the measured time deviation between a recommendation and its consumption by the same user. The values are given in minutes (averaged over the course period) and range from 16,390 minutes (about 11 days) up to 21,537 minutes (about 15 days) for a total course period of 110 days. The Top-3 setting, which performs best for precision and worst for recall, performs best for the timeliness measure (except for $b = 15$). The learning paths with one or two branches ($b = 1$ and $b = 2$) show almost equal and overall the best results for the timeliness deviation. In general, the more branches, the lower the time-dependent accuracy. The same applies to the number of Top-N items.

As precision is more important than timeliness, the approaches with $b = 10$ (which show the highest precision) are presented for each week split. The Top-3 setting starts with less precision compared to the other N settings (see the left side of Figure 7.14 for the already known weekly progress of the results). However, with the beginning of week 5, $N = 3$ outperforms the other settings.

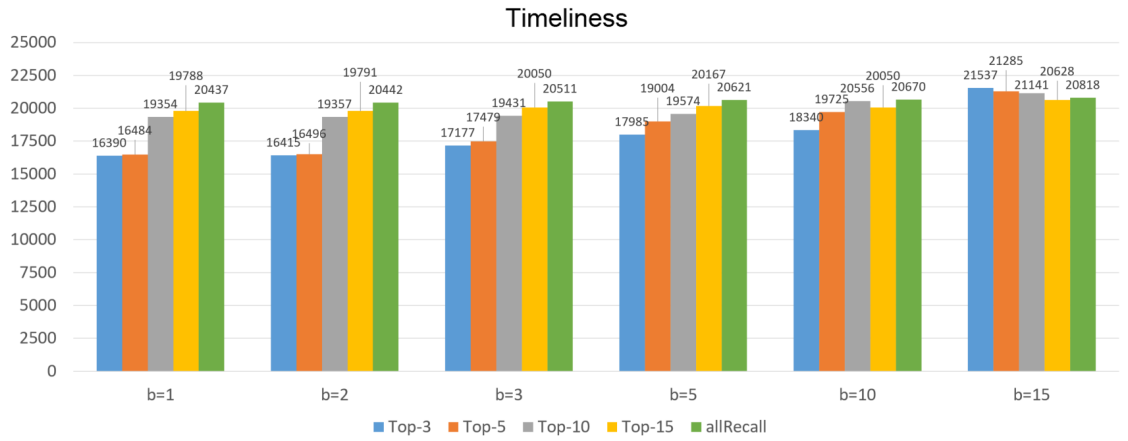

Figure 7.13.: Timeliness deviation for different settings of the Top-N path recommendations.

The right side of Figure 7.14 shows the visualization of the cleaned timeliness deviation over the course period for different Top-N settings. The timeliness decreases dramatically from the course start (ca. 31,000 to 44,000 minutes) to the course end with a cleaned *MATD* of about one day (between 1,000 and 2,000 minutes). All approaches show a stable trend. Thereby, Top-3 and Top-5 recommendations are, except for the beginning, quite similar. The strategy of learning path recommendations improves over time and works exceptionally well in the second half of the course.

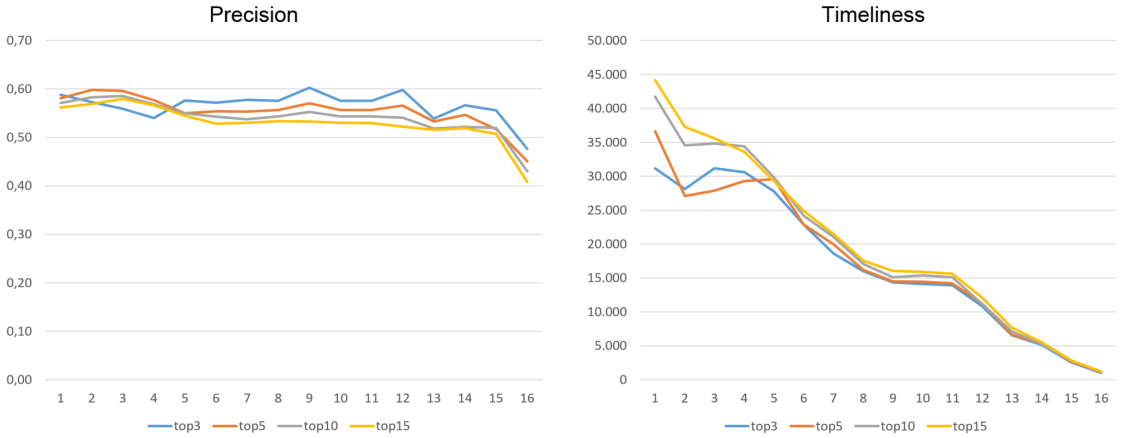

Figure 7.14.: Evaluation results for learning path recommendations ($b = 10$); left: average precision; right: Timeliness values for averaged $MATD_{cleaned}$ given in minutes.

Conclusions

A predicted learning path always contains all of the not-yet-studied items. When a learner strictly follows the recommended learning path, the learner will have consumed all items by the end of the course – without any item being skipped. For this reason, the recall value can be neglected, since it only shows the coverage of relevant items in the current Top-N list. When considering only precision and timeliness, a setting with Top-3 learning recommendations seems to be the best choice.

The decision for the best performing number of branches, in contrast, is not that easy: while $b = 10$ shows the highest precision, the timeliness for that setting performs poorly. The opposite applies for $b = 1$ and $b = 2$. As the precision value is more important for the evaluation and the timeliness is only measured for recommended and relevant items covered by the precision value, the setting with $b = 10$ branches is preferred.

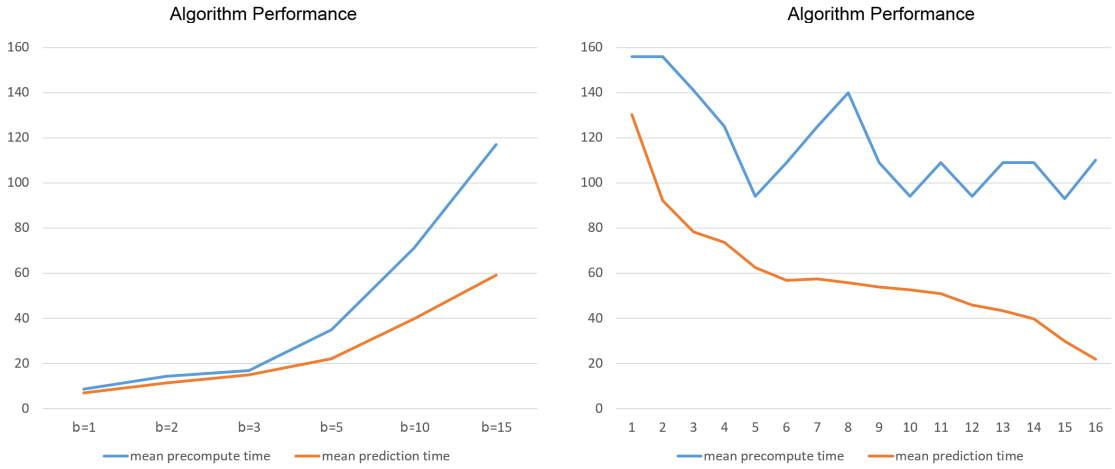

Figure 7.15.: Computional time in milliseconds for the pre-computation step (blue) that is only calculated once per course and the prediction step (orange) that is calculated for each user-request (left: per n-setting; right: over the course period for b=15).

Performance

Figure 7.15 indicates the performance, in terms of calculation time, for different branching settings from $b = 1$ to $b = 15$ (on the left) and over the course period for the most expensive computation ($b = 15$ on the right)[129]. As one can see, especially the pre-computational for generating the knowledge graph shows excellent results, as this step is only required once. However, the prediction step is even faster but must be performed for every user request separately. That makes it in sum a higher value. With about 130 milliseconds prediction time in the worst case, it can be still performed online which makes it the fastest algorithm of the comparison.

7.5. Discussion of the CF Evaluation

Precision & Timeliness

Figure 7.16 shows the precision and timeliness for all algorithms evaluated in this chapter. For a better comparability (and because it shows the best average results), the Top-3 item list setting is chosen for all approaches. The optimized *TBRA* is neglected as only a few users would receive recommendations and thus it distorts the comparison.

At all, the baseline algorithm Slope One on *Learning Objects* performs worst on average over all weeks for precision (0.23) and timeliness (34,083 min) which is not surprising considering the

[129]The evaluation was performed on an Intel Core i7 Quad-Core Processor with 4x 2,6 GHz and 16 GB RAM.

sparse matrix due to the low average number of accesses per item. Indeed, the utilization of higher-level *Learning Units* for the Slope One algorithm brings much better precision (0.46) and timeliness (26,648 min). The incorporation of time weights and the consideration of the already consumed items improves the precision by only 2.6% (by degrading the timeliness) and, thus, is not presented in the figure. The *TBRA* shows a better precision (0.51) than the Slope One approaches but with 36,852 min the worst timeliness in total.

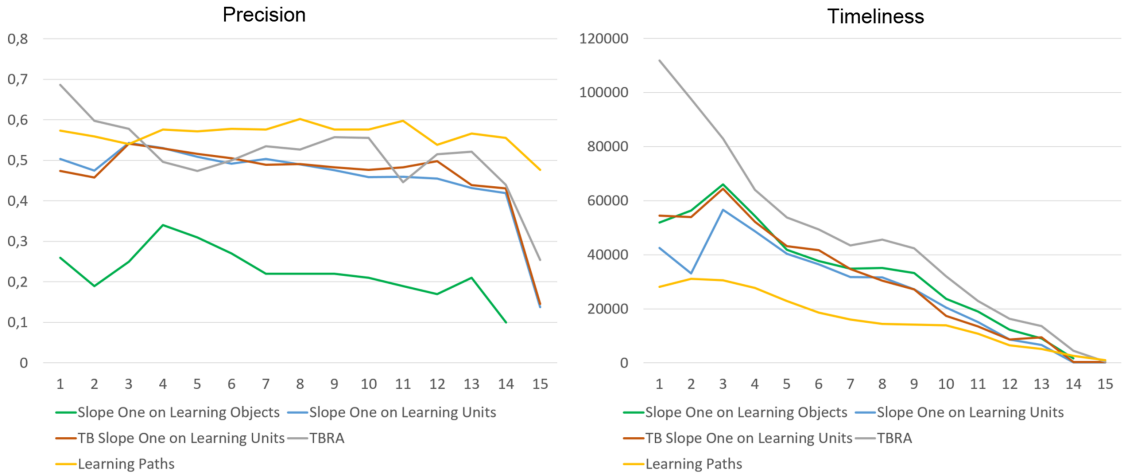

Figure 7.16.: Comparison of the evaluated approaches (each given in the best setting as discussed before); left: average precision; right: Timeliness values for averaged $MATD_{cleaned}$ given in minutes.

The prediction of Learning Paths based on the activities of others performs very well regarding precision with a value of 0.56: More than the half of Top-N recommendations are directly relevant to the learning process. In particular, the timeliness of the recommendations shows outstanding results as it outperforms all other *Collaborative Filtering* algorithms for almost every weekly split – except for the beginning. The *Mean Absolute Timeliness Deviation* is 18,340 minutes. The most significant strength comes from the cold start phase, where the predicted learning paths rely on *Learning Object* metadata that have been entered by educational staff. Thus, even at the beginning of the course, the recommendations are relevant in respect of timeliness. The disadvantage comes from its static concept where every item has to be studied exactly once. However, the generation of learning paths show the best results in this evaluation.

Repetitions

One reason why the approaches reach precision values of only less than 0.70 lies in the composition of the activity data: The number of unique *Learning Units* accessed per user per week as well as the ratio of repeated items is presented in Appendix B.46. Especially for *Self-Regulated Learning* phases (e.g., during the Christmas Holidays and at the end of the course) the repetition rates are higher than during the actual lecture times. On average about 34% of all ”initialized” *xAPI* statements refer to items that have been accessed previously. Thus, an approach that focuses on

the recommendation of items only once per course, such as the recommendation of learning paths, is likely to show a reduced precision of this 34% of repetitions. Moreover, the data even indicate that the students usually did not study all available items. On average the users studied only 34 of the available 106 *Learning Units*. Especially, items at the beginning (e.g., for introduction and motivation purposes) are often skipped by learners – but frequently recommended by the algorithms.

This chapter evaluated different educational *Collaborative Filtering* approaches on the collected activity data and, thereby, raised some issues, such as item repetitions and skipping. Those aspects are taken into account for the *Smart Learning Recommender* that is presented and evaluated in the next chapter.

Next Chapter

8. Design and Evaluation of the Smart Learning Recommender

About this Chapter

The algorithm of the present chapter incorporates rich activity data which results in a novel *Recommender Systems* approach. Thereby, the entire set of gained *xAPI* statements is transferred into the learner's user profile which then helps to model complex knowledge levels at different hierarchy levels of items. The idea is to compensate for the typical drawbacks of *Collaborative Filtering*, such as the cold-start phase, with more detailed data, and to better adapt the recommendations to the user's needs. For reasons of simplicity, this context-aware time-dependent knowledge-based recommender algorithm is called *Smart Learning Recommender (SLR)*.

Additional Behavior Data

The previously analyzed *CF* algorithms utilize just one type of user activities: "initialized" *xAPI* statements representing the user's item accesses. However, these data make up only 47% of the collected *xAPI* statements (see Section 5.2.3). The remaining data give valuable insights into "exited" events, "answered" questions and provided self-assessments as well as "downloaded" files. Taking this information into account, the *Smart Learning Recommender* additionally determines information, for instance, on the duration of content access, subjective assessments of the learners' knowledge levels and verified knowledge levels when users have answered tests and quizzes.

Item Metadata

The content creators and teachers provide a lot of metadata on the items when creating *IMS*-compliant *LOM*, *QTI* and *CC* elements. Among others, the data comprise attributes on *LOs* that are marked as prerequisites (as also considered by the learning paths algorithm), information on the exam relevance of an item and the typical learning time that is given by educational staff. The *Smart Learning Recommender* additionally incorporates the date and time of the lecture when the topics are presented.

Specialized Models

Based on this exhaustive set of information, the *SLR* aims at predicting personal learning progresses within a course. Thereby, additional models help to approximate the real knowledge levels – for instance by incorporating aspects of forgetting.

8.1. Time-dependent Learning Need

As already mentioned, an educational *Recommender System* should aim at identifying the learning needs of a user u for an item i. The learning need is a score that takes the currently

approximated knowledge levels and other related properties (time of the lecture, prerequisites, etc.) as well as the required amount of knowledge into account.

> "The user-item-pair is presented by a relevance score $rscore_{u,i}$ having the value from 0 to 1, where 0 indicates the lowest relevance and 1 indicates the highest possible relevance. The relevance score defines a time and context dependent value and is expressed as a time dependent function:
>
> $$rscore_{u,i,t} = rf_{u,i}(t) \tag{8.1}$$
>
> The relevance function $rf_{u,i}(t)$ of user u for item i is derived from several sub-functions $rf_{u,i,x}(t)$ of individual factors $x_1, ..., x_n$, as a function of time t, each representing another context"
>
> Cf. [161][130].

The sub-functions are based on the previously mentioned item and activity metadata. It turns out that the representation of user feedback as *xAPI* statements and the exhaustive *LO* metadata structure are too inefficient for a direct processing through recommender algorithms. Due to the given complexity of the data structure and the requirement to quickly process this information, the required data are transferred to a proprietary format and retained in a relational database (MySQL). For the sake of clarity, the following paper excerpt introduce the generated factors. A detailed description is presented in the subsequent sections. The sub-functions and the resulting relevance scores are based on real user–item value pairs and are indexed with x:

$$rscore_{u,i,t,x} = rf_{u,i,x}(t). \tag{8.2}$$

> "Since the real learning need changes continuously over time, the factor can be abstracted as a continuous function, as well. The different factor types and considered formulas per user, item and time are:
>
> 1. Interaction with a learning object: This factor indicates how much of the available material for a *Learning Object* was accessed by a student [...]
> 2. Processing time of a *Learning Object*: This factor indicates how long the student learned a *Learning Object*. It is 0 when the student needed exactly the intended time and between 0 and 1 if he[/she] needs more or less time as defined in the metadata [...]
> 3. Self-assessments for this *Learning Object*: A student can explicitly define his[/her] knowledge level at particular points in time on a 1 to 5 stars scale [...]
> 4. Performance in exercises: The percentage of wrong answered questions represents the

[130] The paper has been exclusively written by Christopher Krauss.

relevance of the exercise factor [...]

5. Fulfilled prerequisites: The more a student learned the underlying *Learning Objects*, the higher the relevance score of the subsequent items [...]
6. The lecture times factor indicates the timely relevance of a *Learning Object* for face-to-face lectures [...]
7. Exam relevance: *Learning Objects* that are more relevant for exams show a higher relevance score than optional contents – in terms of a constant value defined in the *Learning Resource Meta-data Specification*.
8. Forgetting effect: After learning an object, the gained knowledge will slowly decrease over time. After each learning iteration, the forgetting factor is set to 0 and [then slowly increases again.]
9. Collaborative learning needs: The relevance functions of similar users on this *Learning Object* are taken into account in order to offset underestimations and bad learning plannings for the current user"

Cf. [161][131].

All factors are in the range of [0,1] and represent the learning need. While some of these factors can be directly influenced or adjusted through interactions with items – e.g., a self-assessment through a rating, the interaction with an item or by giving an answer in an exercise – others are modified by educational staff, such as lecture times or exam relevance. Finally, implicit factors can only be partially affected, these include the collaborative learning need as well as the forgetting effect.

8.2. Calculation of Top-N Recommendations

Before the actual context factors are presented in detail, the mechanism of relevance score calculation is introduced as well as the generation of the Top-N list. This enables an overview of the whole system and should allow for a better understanding of the impact and the inter-dependency of single context factors.

"At the end, all single-factor functions are weighted. The weighted average of all factors describes the total learning need of the *Learning Object* [or other items] for that user and is calculated as

$$rf_{u,i}(t) = \frac{\sum_{x=1}^{n}(w_x * rf_{u,i,x}(t))}{\sum_{x=1}^{n} w_x} \tag{8.3}$$

Here w_x is the weight of a single factor x in $\{x_1, ..., x_n\}$ and n is the number of factors"

[131] The paper has been exclusively written by Christopher Krauss.

Cf. [161][132].

At the beginning of the *Smart Learning Recommender*, the weights were set to a value of 1 in order to have a starting point from which to present recommendations to learners. However, during the development phase, different settings of the weights have been evaluated to determine the most appropriate overall relevance score. These evaluations are subject to further optimization and are presented after a detailed description of the contextual factors in the next section.

While the *Smart Learning Recommender* typically processes a fixed number n of factors $x1, ..., xn$, not all factors are relevant at every point in time. For instance, if users did not access any item, the factors on interaction and processing time are disregarded for the weighting of the overall learning need. In some situations, some content never allow for the incorporation of a particular factor. For example, when an item does not offer any *QTI* assessments, that exercise factor must not be taken into account. In this case, the number n is reduced by the number of factors not considered.

"The overall recommendation engine analyses the current learning need values of the requesting student for all contents in the course. Thereby, the model of the *SLR* can be created offline: The relevance functions are computed at regular intervals by processing all existing user-item-time-triplets. When a user requests recommendations, the relevance scores for all items are calculated on demand by considering the current time value for t. Afterwards, the items are sorted by their learning need value. The result is a Top-N list of *Learning Objects* beginning with the highest relevance scores that represent the most important topics for the student that needs to be learned at that time"

Cf. [161][133].

The overall Top-N recommendations according to $rf_{u,i}(t)$ as well as the contextual recommendations presenting the unique factors $rf_{u,i,x}(t)$ are presented separately. Appendix B.47 briefly introduces the presentation of the recommendations in the *Learning Companion Application*.

"*Learning Objects* are stored in a multi-level hierarchy in order to topically structure a course – top-level items represent a container with a set of sub-level items. [The] user may provide item feedback, in terms of self-assessments, exercises, interactions and processing time, on all hierarchy levels – even for the same topic. This differentiation allows a representation of diverging knowledge levels for top- and sub-level items – e.g. a student might have a good high-level understanding of a [*Learning Unit*], but misses some details on specific [*Learning Objects*] or vice versa. In case the user has not provided the same type of feedback on the item's parent before, it is implicitly transferred from the child to the parent object. So, the

[132]The paper has been exclusively written by Christopher Krauss.
[133]This paper excerpt has been exclusively written by Christopher Krauss.

parent may implicitly represent the average of all child *Learning Objects*.

Hence, the engine needs to avoid recommending the same topic with different detail levels within the predicted list. An algorithm iterates over the generated Top-N list, beginning with the most relevant [item]. An item will be eliminated from the list, in case a related child or parent [item] that describes the same topic, was recommended before and thus, shows a higher score. As a result, students will get recommendations for all topics of a course in a predicted order, but only on an appropriate detail level"

Cf. [161][134].

The Top-N list might comprise *Learning Units* or low-level *Learning Objects* which belong to a different *Learning Unit*. A previously listed item is not separately presented at another level for the same topic in the Top-N list to avoid redundancy. Usually, items at lower levels are more frequently recommended, because learners access some items at a certain level and thus the other items of this level show a higher learning need than their parent items.

8.3. Context-Factors

The contextual factors introduced in this work are not complete. They represent only the main classes of different context-sensitive influences which help to identify knowledge levels, knowledge gaps and appropriate recommendations. A great number of possible inputs (such as assessments at the course start or questions on individual motivations) are missing here because they are not covered by the collected data. However, new context factors can be easily added by following the context factor definition, where every factor represents the learning need of a user for an item at a specific point in time.

The utilized context factors represent the different dimensions of a *Context-Aware Recommender System*. They can either be classified as "User", "Item" or "Activity" data according to the context categories of Adomavicius and Tuzhilin [10] as well as Verbert et al. [270, p. 322 - 324]. However, a strict differentiation seems to be hard, as all of them incorporate at least a second context dimension: the progress over time.

8.3.1. Self-Assessments

When a learner starts to use the *Learning Companion Application*, the learner can provide self-assessments relating to the items. A self-assessment is given on a one-to-five star scale (zero when there is no self-assessment), where one represents no knowledge and five stands for expert knowledge. The rating can be given by clicking on the according star, as is familiar from movie or

[134] The paper has been exclusively written by Christopher Krauss.

e-commerce services. The star assessments are available for the hierarchy levels of *Learning Units* and modules (parent of an *Learning Unit*). Moreover, even the learning goals (given in the *LOs* metadata) can be rated at the beginning of a *Learning Unit* and a second time at the end (see Appendix B.48). Moreover, the learner can visualize the history of their self-assessments with the help of the statistics view which also helps to analyze the learner's perception of the knowledge transfer. The *Learning Companion Application* encourages learners to provide a lot of personal information regarding the offered item at various points in time – the more, the better. Thereby, the following equation applies to the self-assessment factor $x1$:

$$rscore_{u,i,t,x1} = 1 - \frac{currentKnowledgeLevel_{u,i,t}}{highestKnowledgeLevel}. \tag{8.4}$$

Figure 8.1 shows the possible range of learning needs that reflect the self-assessments. The progress in this chart does not depend on time but on the provided feedback value. The higher the self-assessment (e.g., a learner is rated as an expert), the lower the learning need. Consequently, a recommendation of a high rated item is less probable.

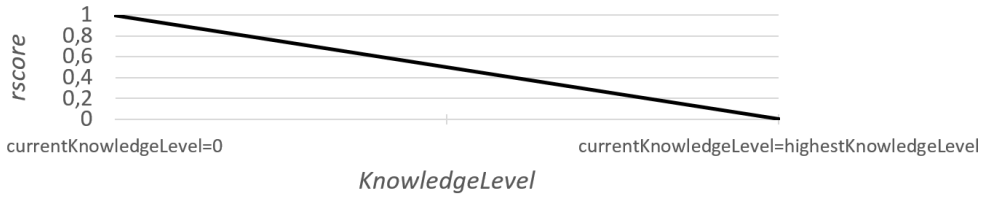

Figure 8.1.: *Smart Learning Recommender*: Range of relevance values for self-assessments.

A self-assessment is especially helpful if the learner did not provide any other feedback for that item – for instance, when the learner neither accessed the *Learning Unit* nor answered any test questions relating to it. The value reflects the learner's assumption regarding their existing knowledge in that area from reading the headline or the related learning goal at the beginning of a unit. Where the learner did not provide any other assessment feedback, the item level above automatically represents an average of the assessments of its child items.

8.3.2. Interactions with Items

A more valuable feedback than just self-assessment is the information on item accesses – as already presented in the evaluation of the *Collaborative Filtering* approaches. When users access items, they potentially consume the offered media (e.g., a text, an animation or a video). Alternatively, when learners did not provide any additional feedback and not access any items, the algorithm assumes the user has a lack of knowledge on that topic.

The interaction or access factor $x2$ indicates how much of the available material $availableContent_i$

for an item i is accessed by a student u at a specific moment in time t:

$$rscore_{u,i,t,x2} = 1 - \frac{accessedContent_{u,i,t}}{availableContent_i}. \tag{8.5}$$

This can be the percentage of a watched video or audio item as well as how much the student scrolled through a text. Figure 8.2 presents the value range for the relevance score.

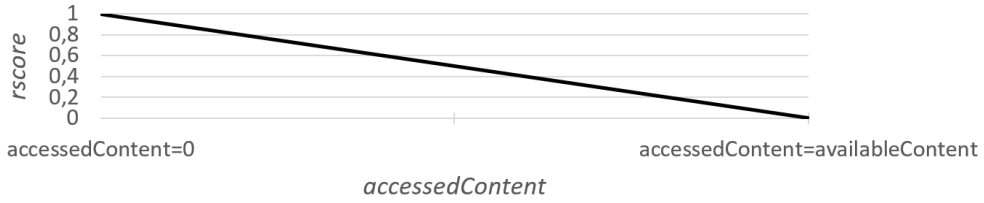

Figure 8.2.: *Smart Learning Recommender*: Range of relevance values for interactions.

This factor represents the number of accesses and is also transferred to higher levels of the item hierarchy. For instance if only one out of four *Learning Objects* have been opened, the relevance score of the parent *Learning Unit* is 0.75 as three quarters are then missing. The more *Learning Objects* have been accessed within a *Learning Unit*, the lower is the need for learning that particular unit.

8.3.3. Processing Time of an Item

While the interaction with an item only indicates if it has been opened, it does not directly reflect that the user has indeed read or consumed the content. The user of the *LCA* might, for instance, click on an *LO* and directly close it again, as the learner had expected something else. Therefore, the *Smart Learning Recommender* measures the time between the ”initialized” event and the ”exited” or the ”abandoned” *xAPI* statement. This timespan represents the time between accessing and leaving the content. A similar data foundation was used by Yi et al. to predict results of a search engine based on dwell times (time spend on a website) instead of page clicks [295]. Thereby, Yi et al. improved the mean precision by about 1% compared to the traditional click-based approach.

Aronson et al. define three categories of time in learning settings: ”engaged time”, ”instructional and non-instructional time” and ”academic learning time” [20] (also cf. [105]). The ”engaged time” can be stopped using the *xAPI* statements and represents the time the user engages with the content via the *LCA*. Allocated ”instructional time” represents the intended time of learning at the educational institution and the ”non-instructional time” stands for the time the user is expected to spend for *Self-Regulated Learning*. The latter corresponds to a definition of the typical required learning time as set by educational staff. In the Smart Learning project, content creators defined the typical learning time for each *LO*. The learning time for a high-level item is the sum

of the time of all of the low-level items contained within it. Finally, the "academic learning time" represents the real timespan of knowledge acquisition, which is often smaller than the "engaged time". In a typical *Personal Learning Environment*, the "engaged time" is hard to measure, as the user might be distracted and not directly engage with the content – even though the learner opened that item.

However, the idea of the processing time factor is to relate the "engaged time" of the learner with the actual "instructional time" given by the educational staff. The closer the "engaged time" is to the intended "instructional time", the more probable the learner processed the content as intended.

The processing time factor $x3$ indicates how long the learners studied items. It is 0 when the learners spend more or exactly the intended time $timeIntended_i$ and it is between 0 and 1 if the spend time $timeNeeded_{u,i,t}$ of user u on item i at time t is less time than defined in the metadata.

$$rscore_{u,i,t,x3} = \begin{cases} 1 - \frac{timeNeeded_{u,i,t}}{timeIntended_i} & , \text{ if } timeNeeded_{u,i,t} \leq timeIntended_i \\ 0 & , \text{ if } timeNeeded_{u,i,t} > timeIntended_i \end{cases}. \tag{8.6}$$

The formula requires that the instructional time $timeIntended_i$ is not 0. Figure 8.3 visualizes the range of relevance values for $rscore_{u,i,t,x3}$.

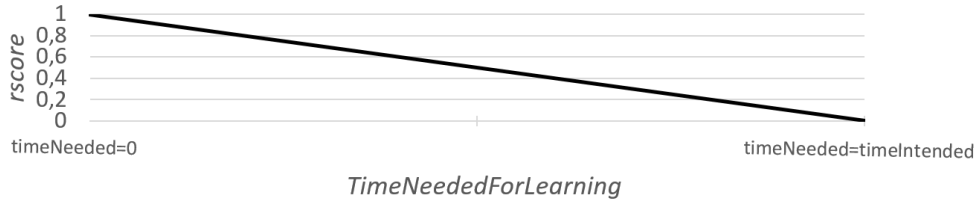

Figure 8.3.: *Smart Learning Recommender*: Range of relevance values for processing time.

In an initial phase, the processing time is penalized when learners spend more than the intended time for studying – as presented in Appendix B.49. However, feedback of learners led to an adjustment of the formula that better represents the users' expectations.

This factor does not take the personal learning speed into account, as one learner might have a slow and another learner might have a fast speed of comprehension. However, as the individual learning speed might be relatively constant compared to the intended instructional time, the factor values still give a good indication which items require more time of engagement of the same user.

8.3.4. Performance in Exercises

The latter three factors indicate how much users think they know (in terms of self-assessments) or how much they interacted with the items. However, these factors do not yield information regarding the level of comprehension of the studied topics presented through the items.

Therefore, researchers note the importance of analyzing exercise performance [208, 204, 16]. The Smart Learning project offers different types of *QTI* exercises. First and foremost, multiple-choice tests (with only one correct answer each) are offered in the *AWT* course where learners need to answer questions regarding previously learned topics. Independently from the actual type of exercise, the results are stored as a value between 0 and 1. Thereby, the learning need given with the exercise factor $x4$ is the inverse of the determined knowledge in the exercises. Thus, the percentage of wrongly answered questions $wrongAnsweredQ_{u,i,t}$ represents the relevance of the exercise factor:

$$rscore_{u,i,t,x4} = 1 - \frac{rightAnsweredQ_{u,i,t}}{allQuestions_i}. \tag{8.7}$$

Equation (8.7) is the same as $\frac{wrongAnsweredQ_{u,i,t}}{allQuestions_i}$. Questions that are skipped will be treated as having wrong answers, so that the relevance score is only 0, when a student correctly answered all questions. Figure 8.4 presents the value range of $rscore_{u,i,t,x4}$.

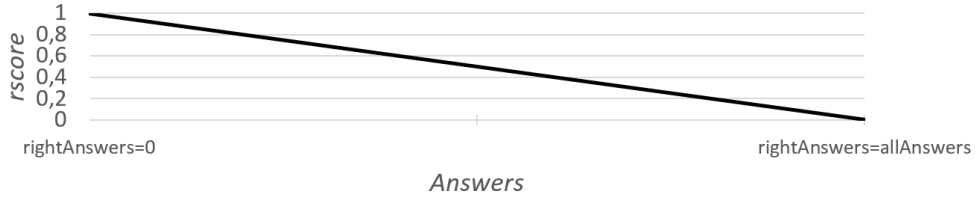

Figure 8.4.: *Smart Learning Recommender*: Range of relevance values for exercises.

If there is no question in an *LO*, this factor is not considered for the overall learning need weighting. If more than one *QTI* assessment is presented in a *Learning Object*, the relevance scores are averaged. Moreover, the resulting relevance scores on the *Learning Object* level are transferred to the parent *Learning Unit* also. Thereby, the relevance scores of all lower-level items are averaged. This allows the user's comprehension to be observed (in terms of exercise results) even for higher item levels.

8.3.5. Fulfilled Prerequisites

Besides the activity information collected through direct interactions with learning items, also instructional metadata play crucial roles for the *SLR*. Besides their hierarchical structure, contents often show a didactic structure that is given with the prerequisite setting in the *LOM* specification. Transferred to the concept of the learning need value, an item is more relevant at time t when all prerequisite items have been studied. Therefore, this factor relies on the interaction factors of the prerequisite items.

The more a student accessed (according to factor $x2$) the underlying learning objects, the higher

the relevance score of the subsequent items for the prerequisites factor $x5$ is:

$$rscore_{u,i,t,x5} = \frac{fulfilledPreRequisites_{u,i,t}}{allPreRequisites_i}. \tag{8.8}$$

In case there is no prerequisite for a learning object, the relevance score is 1. Figure 8.5 presents the value range of factor $x5$.

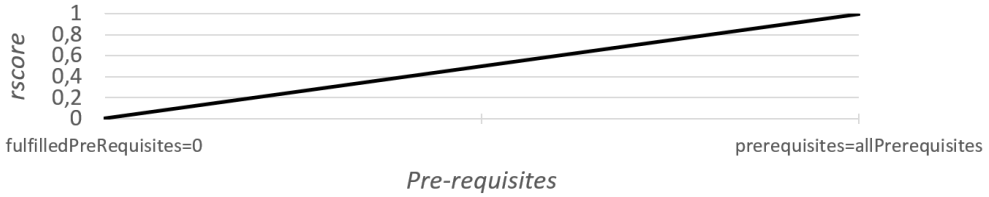

Figure 8.5.: *Smart Learning Recommender*: Range of relevance values for fulfilled prerequisites.

This factor has a similar meaning as a didactic learning path – presented in the previous chapter. However, a path is not directly presented to the learner but rather plays a role in the calculation of recommendations.

8.3.6. Lecture times

While the prerequisites factor supports the *Self-Regulated Learning* skills of learners independently from the given lectures, the actual course schedule may also play a role in the relevance of items. Learners should prepare and wrap-up the presented materials to reach the course goals efficiently.

Thereby, the lecture times factor $x6$ indicates the timely relevance of an item for face-to-face lectures. The closer the lecture at time $timeOfLecture_i$ where item i is presented, the higher is the resulting relevance score:

$$rscore_{u,i,t,x6} = 1 - \left(prepPhaseFactor * \frac{timeOfLecture_i - t}{courseDuration}\right)^2. \tag{8.9}$$

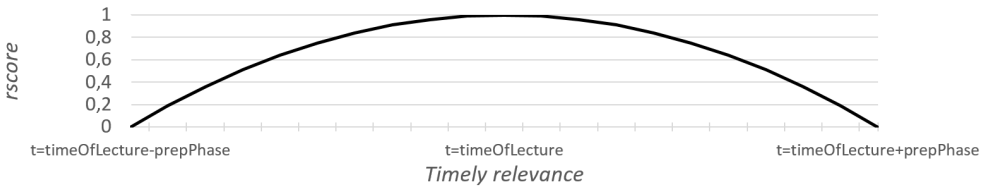

Figure 8.6.: *Smart Learning Recommender*: Range of relevance values for lecture times.

Figure 8.6 shows the range of the relevance scores of factor $x6$. The $courseDuration$ (time of the course ending minus time of the course start) must be higher than 0, and the $timeOfLecture$ must be within the $courseDuration$. The $prepPhaseFactor$ must be defined by the experts and relates to the duration of both the preparation as well as the wrap-up phase of a lecture, where the contents

concerned are more relevant ($prepPhase = \frac{courseDuration}{prepPhaseFactor}$). For instance, for $prepPhaseFactor = 1$ the preparation phase is exactly the course duration, for $prepPhaseFactor = 2$ it is half the course duration, for $prepPhaseFactor = 4$ the preparation phase is one quarter of the course duration and so on. The higher the number is, the later is the start of the preparation phase and, thus, the later this item is recommended.

For the *AWT* course, the $prepPhaseFactor$ is set to 2 and thus shows long preparation and wrap-up phases. This is done in order to recommend lecture materials right from the course start and to better compensate for the cold start problem. Contents are manually allocated to specific lectures by educational staff. The learner can access the list of concerned items presented in each lecture via the course schedule in *LCA*.

In case the current time t does not fall into either the preparation or wrap-up phase, the relevance score is set to 0. When an item is presented in two different lectures, the lecture that is closer to the particular point in time defines the resulting learning need.

8.3.7. Item Relevance for the Course Goal

A parameter that may play an important role in goal-oriented learning is the defined importance of learning that item, which can be adjusted by the educational staff. For instance, for courses with an exam at the end, not all topics are equally relevant. Items that are more relevant for exams show a higher relevance score $rscore_{u,i,t,x7}$ for the goal relevance factor $x7$ than optional contents – expressed by a constant value defined in the learning object metadata.

The score should allow for a differentiation of the importance of different items. If all items show the same increasing relevance, their relative importance is still equal. That is why this factor can be given as a simple constant between 0 and 1 instead of a linear or more complex expression. With the constant value, the higher the value is, the more important is the item for reaching the course goal.

8.3.8. Collaborative Learning Need

Besides the collected personal behavioral data and information provided by educational staff, the interactions of other learners allow for an indication of the item's relevance. For example, the greater is the need for classmates to learn a specific item, the more probable is it that the actual user needs to learn the same item, which is a conventional approach in *Collaborative Filtering* and should also be taken into account as a contextual factor $x8$.

The relevance functions of similar users on this *Learning Object* are taken into account in order to offset underestimations and bad planning for the current user. In an initial phase, the mean average learning need of all other students (but without their collaborative learning need to avoid

recursion loops) for this *Learning Object* represent this factor.

$$rscore_{u,i,t,x8} = \frac{\sum_{y=1}^{|Y|}(rf_{y,i}(t))}{|Y|}, \tag{8.10}$$

where Y is the set of all users of the class without the current user u and $|Y|$ is its cardinality. $rf_{y,i}(t)$ is the learning need value of a relevant factors (without $x8$) for the same item i at time t.

The idea is that the user also receives recommendations based on the determined learning need of other learners which cannot directly be provided or adjusted by the user. For instance, students might believe that they have expert knowledge in a specific area and thus skip the related items. However, when other students detect some learning issues that result in a higher learning need – e.g., an exercise question might be hard to solve – the actual user also gets a recommendation based on the learning weakness of the others.

8.3.9. Human Memory and Forgetting Effect

The most sophisticated factor in the *Smart Learning Recommender* is the model of forgetting [135]. Ebbinghaus started in the late 19th century with the first experiments to model forgetting within the field of experimental psychology [98].

Based on the findings of other researchers [98, 140, 23, 254] the *Smart Learning Recommender* also incorporates the process of forgetting. Therefore, the parameters that influence the knowledge acquisition and preservation are analyzed. A literature review as well as an initial experiment resulted in multiple critical parameters that affect forgetting (apart from time): The type of medium, its difficulty level, prior knowledge of the learner, the learner's interest in the topic, personal memory strength, the time and extent of learning repetitions and even knowledge recall for retention tests. Appendix B.50 introduces these considerations and the experiment that led to the model. The relevance score of the forgetting factor $x9$ is restricted to the range [0,1]. Thereby, the e-function shows the most similar progress of the determined forgetting of the participants in the survey:

$$rscore_{u,i,t,x9} = 1 - e^{-(timeFactor(t) - E_{rep} - E_{ret} + E_m + E_d)} + \alpha. \tag{8.11}$$

Formula 8.11 includes two main parameters: an e-function (with respect to learning times $timeFactor(t)$ and media metadata, E_{rep}, E_{ret}, E_m and E_d) as well as a personalized parameter α that adapts the individual forgetting process to the learner. While α has only been considered in theory, the set of other parameters was analyzed with real participants in a small experiment.

[135] The experiments on transferring existing forgetting models to the area of educational digital media was part of the master thesis of Rakesh Chandru with the title "Oblivion in Recommender Systems – The forgetting effect in predictions". Christopher Krauss supervised the thesis. Rakesh Chandru worked as a student employee at the Smart Learning project led by Christopher Krauss. Rakesh Chandru implemented the algorithm and conducted the experiments on forgetting with real people.

The *timeFactor* represents the forgetting by only taking the time dimension into account:

$$timeFactor(t) = \frac{t - t_i}{courseDuration} * \sqrt{\frac{courseDuration}{durationOfForgetting}}. \tag{8.12}$$

The value $t - t_i$ corresponds to the difference between current time t and the first access time of the learning content t_i. The *courseDuration* (time of the course ending minus time of the course start) must be higher than 0. The duration of forgetting *durationOfForgetting* represents the period in which the content has been totally forgotten (if no other parameter affects it) and is set in an initial version to *courseDuration*/2. This means that during a semester, students forget an item's content after about half of the course period. This value must be adjusted by experts. Where learners did not access an item, the forgetting relevance score is set to 1.

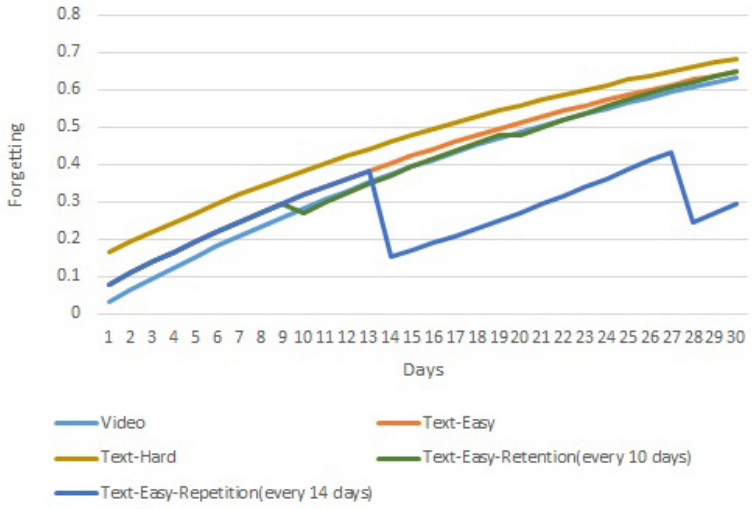

Figure 8.7.: Examples of the mathematical model for predicted forgetting of different media types and with retention tests and repetitions on a 0 to 1 scale; figure by Rakesh Chandru.

E_{rep} is the repetition effect which is affected by another item access, E_{ret} is the retention test effect affected by answering related exercises, E_m is the media type effect and E_d represents the difficulty level. All parameters, E_{rep}, E_{ret}, E_m and E_d, are in the range of [0,1], where a higher value accelerates the forgetting progress and a lower value decelerates it. The details on the parameters are important for the understanding of the forgetting formula and are presented in Appendix B.51 for the sake of clarity.

Figure 8.7 shows example curves for the forgetting factor under different settings of parameters but without the inclusion of a personalization parameter. The chart visualizes the findings of an experiment that resulted in the forgetting model introduced in this work.

8.4. Detection of Appropriate Time-dependent Weights

In order to evaluate the ideal weights for different factors that will lead to the most appropriate recommendations, different methods are analyzed. The first one is a naive approach, where every factor is given equal weight and thus has an equal effect on the overall learning need.

8.4.1. Analysis of Individual Context Factors

To analyze this approach, the effect of each factor is determined independently. All approaches are based on Evaluation Setting 2 (see Appendix B.40) – with the major difference that all available activity statements on *Learning Units* are incorporated. Figure 8.8 shows the results for the *Smart Learning Recommender* on the *AWT* dataset, where only one factor is considered as relevant per iteration. This means, for instance, that the "ProcessingTime" progress visualizes precision and timeliness for the Top-N items that show the highest values of the "ProcessingTime" factor. In other words: only those items are presented to the learner that have been consumed least by this user. The recommendation list based just on the "ForgettingEffect" contains only items that might have been forgotten and the list based just on the "Lectures" factor presents only items that are most relevant for the preparation or wrap-up of lectures. The "OverallLearningNeed" analysis displays additionally the results for a Top-N list that are based on the average, equally weighted, values of all factors.

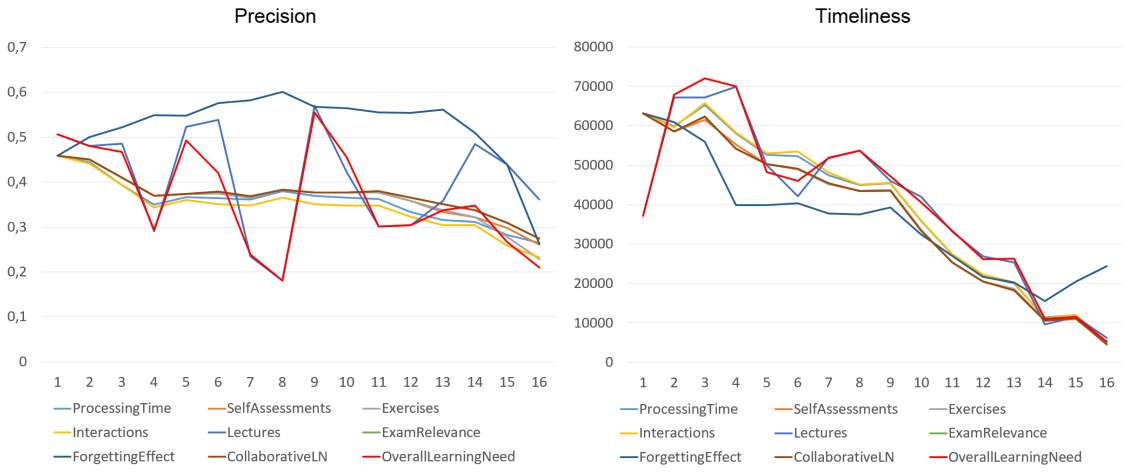

Figure 8.8.: Precision in percent (left) and timeliness in minutes (right) of the *SLR* (only Top-3 recommendations) approach with equal weights.

Single Factor Setting

Except for the forgetting factor and the lecture factor, all other factors show similar results for precision and timeliness, with the interaction factor showing the worst values and the collaborative factor the best. All of these factors start at a precision of 0.46 and end with a precision of about 0.23 to 0.27 (which is very low). The progress of the lecture factor draws a diverging picture with

scattered precision values: little in week 4, 7, 8, 11 as well as week 12 and comparably high in all other weeks. This might be due to the five different teachers who held the 12 face-to-face meetings. The forgetting factor is especially interesting as it shows a distinct progress and ranges for the longest time between 0.50 and 0.60 and then drops off to 0.36 at the end of the course. Also, regarding timeliness, the forgetting factor leads to the most appropriate recommendations[136].

Equal Weights Evaluation

The red line in Figure 8.8 presents the results of the overall learning need that is the average of all considered factors. This means that the weight w_x according to Equation 8.3 for all factors is set to 1. The overall learning need is dramatically influenced by the lecture factor, as this is the only factor that comprises data for every item right from the beginning. The influence of the lecture factor on the overall learning need, however, lessens in the final period (starting in week 12), when the final lecture has been given. The period after this lecture requires *Self-Regulated Learning* skill from the users in order to prepare for the final test. The precision of this approach is very scattered and lies between 0.18 and 0.55. Thereby, the total average precision over the entire course period is only 0.359.

Neglecting Interpolated Data

In a second attempt, the factors were adjusted as follows: all default and approximated values for the factor's relevance scores are removed when there is no real feedback from the user. For example, in the first attempt, the interaction factor showed a relevance score of one until the item was accessed for the first time by the user and then slowly decreased by the percentage of consumed sub-items. In the second attempt, the interaction factor was neglected in the averaging process until the item was accessed for the first time and then slowly decreased, starting with a value of one. Also, the forgetting factor was ignored until the user processed the related item for the first time and the exercise factor was neglected until the users started answering exercise questions instead of having a relevance of one right from the beginning of the course. Only the lecture time and exam relevance factors which are independent of a personal feedback are available to the whole time.

One could argue that this reduces the number of items which might be presented in the Top-N list. Indeed, only those factors are considered which have feedback from the actual user. However, the lecture time factor is present at every point which allows for the recommendation of every item with or without feedback from the user. As long as the user has not opened any topic, item recommendations follow the course schedule implicitly due to the lecture time factor. Figure 8.9 displays a cold start phase where some factors, such as interactions or exercises are missing in the beginning of the course, but it then shows an improvement of the precision of almost all factors. Moreover, the equal weighted learning need is also somewhat more stable and shows a precision of between 0.30 and 0.60 with 0.478 on average.

[136] The reason why the timeliness value of the forgetting effect exceeds the maximum of 10,080 minutes in week 16 can be explained by the fact that some users used the *Learning Companion Application* even after the course had ended. One learner, for instance, resat an exam about two months after the first test and prepared with *Learning Companion Application* for the second test, which influenced the timeliness.

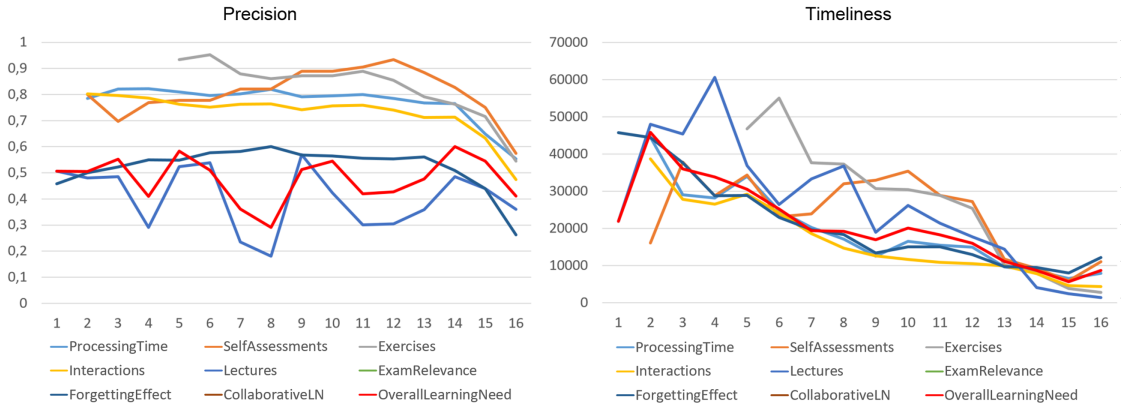

Figure 8.9.: Precision (left) and timeliness (right) of the *SLR* (only Top-3 recommendations) approach for factors with user feedback only.

Unfortunately, the timeliness factor draws a less distinct picture. The cleaned *Mean Absolute Timeliness Deviation* depend heavily on the type of the factor and are much more scattered: sometimes the factors are better, for instance for interactions, and sometimes worse, for example for exercises, compared to the previous attempt. As the precision measure plays a more critical role than the timeliness, it should obtain a higher focus on optimization. This is why the further analysis relies on this second approach.

Factor Precision as Weights

In a first attempt to determine appropriate time-weights, the factor's precision values per week (given in Figure 8.9) are utilized as weight for the same factor. The intuitive approach is that factors with a higher precision of their individual Top-N lists have more influence on the total recommendations process. Additionally, a power β is given to the weights to maximize its effect. Based on the precision $prec_x$, the weight w_x for factor x is calculated as:

$$w_x = (prec_x)^{\beta}. \tag{8.13}$$

The most precise results are reached with a power $\beta = 10$. However, the average resulting precision is only 0.461 and is thus below the precision of equal weights of 0.478 (as presented before).

8.4.2. Determining appropriate Time Weights with Linear Models

In a second phase, particular factor weights are determined that lead to an improvement of, first and foremost, the precision and, secondary, the timeliness. Therefore, only activity data of the training set are processed. These are the z items that user u accessed in the training set Tr. The training and test set splitting follows the increasing time-window specification of Campos et al. [57]. Adjustments to the weighting model are evaluated according to Evaluation Setting 2.

To determine precise weights for the *SLR* model, all historic items' relevance scores of user u are

analyzed. The idea is to allocate a higher weight to those factors that help to differentiate the last accessed items from the rest. In other words: The most important factors of last accessed items should receive higher weights as they should play more significant roles for recommendations in the current situation.

Time Weight

A weight is determined for each factor x, user u and time of recommendation t separately. However, for the evaluation, the factor weights are averaged over all users to obtain a user-independent weight $w_{x,t}$. The most precise model is called the "normalized average factor deviation" approach and is presented in the following formula[137]:

$$w_{x,u,t} = \frac{\sum_{i=1}^{z} \left(decay_{t_i,t} * \Delta rscore_{u,i,t,x}{}^{\beta} * rscore_0 \right)}{\sum_{i=1}^{z} (decay_{t_i,t})}. \tag{8.14}$$

Decay Effect

The z items of the training set are processed. The calculated weight $w_{x,u,t}$ consists of three major parts: a decay effect $decay_{t_i,t}$, a deviation from the average score $\Delta rscore_{u,i,t,x}$ and a parameter $rscore_0$ for normalizing the score. The decay effect $decay_{t_i,t}$ penalizes older activity data and prefers younger information. It is given as:

$$decay_{t_i,t} = \frac{t_{first} - t_i}{t_{first} - t}. \tag{8.15}$$

The decay effect is important because the relevance values change over time according to the special educational *Recommender System* paradigm. Thus, the decay value is higher, the younger the relevance scores at time t_i. For instance for the interaction factor, the closer the last item access is to the current point in time t, the more relevant is this data point for the calculation. t_{first} is the first date on which an item access is considered for the weighting (e.g., the course start).

Score Deviation

$\Delta rscore_{u,i,t,x}$ is the deviation of a particular factor value $rscore_{u,i,t,x}$ from the average value $\varnothing rscore_{u,t,x}$. Thereby, $\varnothing rscore_{u,t,x}$ is the average relevance score of all items depending on the user, the time and the factor. $\Delta rscore_{u,i,t,x}$ is expressed as:

$$\Delta rscore_{u,i,t,x} = \begin{cases} rscore_{u,i,t,x} - \varnothing rscore_{u,t,x} & , \text{ if } rscore_{u,i,t,x} > \varnothing rscore_{u,t,x} \\ 0 & , \text{ if } rscore_{u,i,t,x} \leq \varnothing rscore_{u,t,x} \end{cases}. \tag{8.16}$$

Only positive values for the deviation are allowed in order to focus on only positive effects of the $\Delta rscore_{u,i,t,x}$. If the values were below 0, the consideration of this item would allow for negative factor weights.

The value β adjusts the power of the relevance value $rscore_{u,i,t,x}$. This means the higher β, the more scattered are the single factor weights. High factor values have an even higher impact (with

[137] The history of different experiments to determine the most precise weights for the model is presented in Appendix B.52.

a high value of β) and low factor values have a lesser effect.

Score Normalization

$rscore_0$ normalizes the weight values according to the average. As a result, a relative average deviation is calculated instead of an absolute one. By doing this, a factor that normally shows lower average values has a similar influence on the overall recommendation process as the other factors, and the effect of factors with high values and just small deviations is alleviated.

$$rscore_0 = \frac{1}{\varnothing rscore_{u,t,x}}. \tag{8.17}$$

If, in an unrealistic case, the average factor value $\varnothing rscore_{u,t,x}$ is zero (which would mean that the factor values of all items indicate that the topic does not have to be learned at all), $rscore_0$ is set to 1.

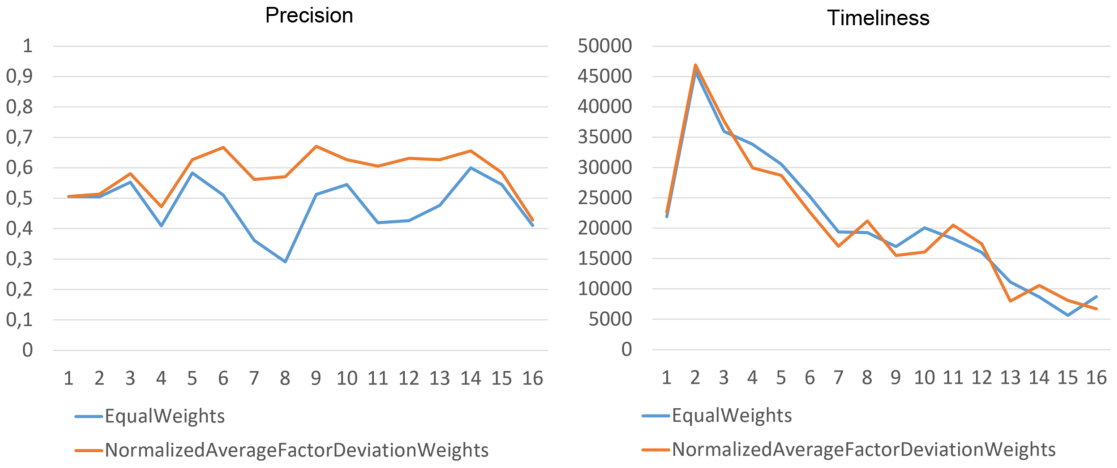

Figure 8.10.: Evaluation of the accuracy of *SLR* through the weight detection with a linear model; left: average precision; right: Timeliness values for averaged $MATD_{cleaned}$ given in minutes.

Figure 8.10 presents the improvements of the linear model compared to the original equal weighted model. The "normalized average factor deviation" reached an average precision of 0.583 and, thus, is about 10.5% more precise than the equal weighted model. The timeliness, in turn, is almost equal and improves by only about 8 hours on average (from 21,100 minutes to 20,607 minutes).

8.4.3. Determining appropriate Time Weights with Artificial Neural Networks

Finally, the determination of time weights is additionally realized with the help of a totally different approach: by utilizing an *Artificial Neural Network (ANN)*. Therefore, RapidMiner[138] is

[138]RapidMiner. See: `https://rapidminer.com/` (Accessed: 03.11.2017).

applied with a convolutional neural network[139] that contains just one hidden layer. A separate model is trained for each week split t and the evaluation setting still follows the definition given in the Appendix B.40 with activities on *Learning Units* in the *AWT* course.

One input vector represents a user–item–time triplet. Figure 8.11 presents the inner composition of a single node k (a neuron of the *ANN*). An *ANN* consists of one or more hidden layers – each layer, in turn, contains a set of nodes. An input vector comprises $n+1$ input features which are the n factor relevance values of the *SLR* (per user, item and time). Additionally, the first input value per node is set to a value of one, which makes it a node-specific bias. The node's output o_k is the predicted value (e.g., the overall learning need) for the user–item–time triplet. The training values for the output are 1 if this item i has been accessed by the user u in the last week before the split t and 0 if not.

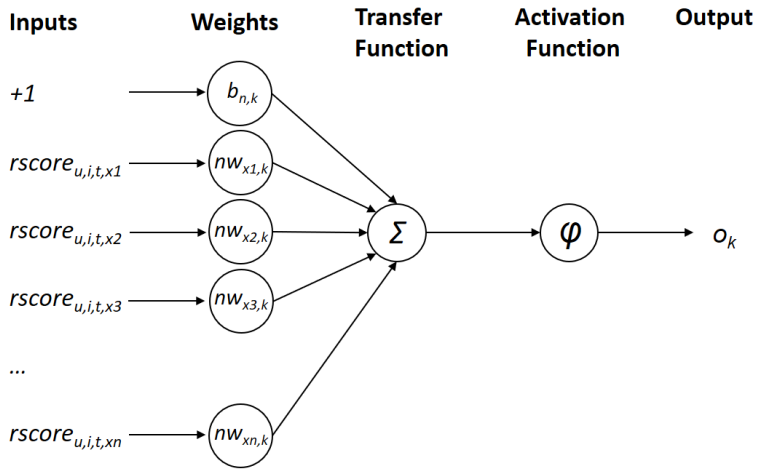

Figure 8.11.: Composition of a single node k in an *ANN* that is trained with the factor's relevance values as input.

The node's output is calculated as:

$$o_k = \varphi((\sum_{x=1}^{n} rscore_{u,i,t,x} * nw_{x,k}) + b_k). \tag{8.18}$$

In contrast to the actual intention of an *Artificial Neural Network*, where the predicted output values o_k are of interest, the approach in this work targets the determined weights of the trained nodes. Thereby, each of the nodes contains a weight vector $\overrightarrow{NW}_k$. This weight vector has the same number $n+1$ of dimensions as the *ANN* has as input features (where the first weight represents

[139] RapidMiner Neural Nets. See `https://docs.rapidminer.com/studio/operators/modeling/predictive/neural_nets/neural_net.html` (Accessed: 03.11.2017).

the bias b_k). Thus, each weight vector $\overrightarrow{NW}_k$ contains the following values:

$$\overrightarrow{NW}_k = \begin{pmatrix} b_k \\ nw_{x1,k} \\ nw_{x2,k} \\ nw_{x3,k} \\ ... \\ nw_{xn,k} \end{pmatrix}. \tag{8.19}$$

The activation function φ is a sigmoid function which is common for *ANNs* that aim at solving linear problems.

$$\varphi(m) = \frac{1}{1 + e^{-m}}. \tag{8.20}$$

Since only one hidden layer is utilized, each factor-specific input feature ($rscore_{u,i,t,x1}$, ..., $rscore_{u,i,t,xn}$) is directly related to a weight of node k ($nw_{x1,k}$, ..., $nw_{xn,k}$). The fact that the node's weights would lose a direct relation to the factor-specific input scores results in the necessity to utilize only a first hidden layer. As a consequence, the weights fit the requirements of the *SLR* weighting formula presented in Formula 8.3. If the hidden layer contains more than one node, the weights per factor x are averaged according to the node's bias b_k which results in a final weight w_x of factor x:

$$w_x = \frac{\sum_{k=1}^{K} nw_{x,k} * b_k}{\sum_{k=1}^{K} b_k}. \tag{8.21}$$

K is the number of all nodes in the hidden layer. The determined weight w_x for each factor is subsequently applied to the factor weight algorithm of the *Smart Learning Recommender* given in Formula 8.3. The rest of the evaluation has been conducted as known from Evaluation Setting 2.

Figure 8.12 shows the different precision and timeliness results for varying numbers of nodes in the hidden layer. With a sample of 50 nodes, the *ANN* reaches the highest average precision values of 0.587 and, thus, is 0.4% higher than the precision of the linear model presented before[140]. The resulting cleaned MATD, in turn, is slightly worse with 20,811 minutes (compared to the 20,607 minutes of the linear model).

8.5. Analysis of the Smart Learning Recommender

Performance

This last section of the chapter analyzes the introduced *Smart Learning Recommender* algorithm. Figure 8.13 shows the average calculation time for offline data preprocessing and the online

[140] An attempt to directly predict the learning need value instead of determining time weights as an intermediate value results in an average precision of only 0.354. The *ANN* uses the same kind of input values, with two hidden layers and 50 nodes.

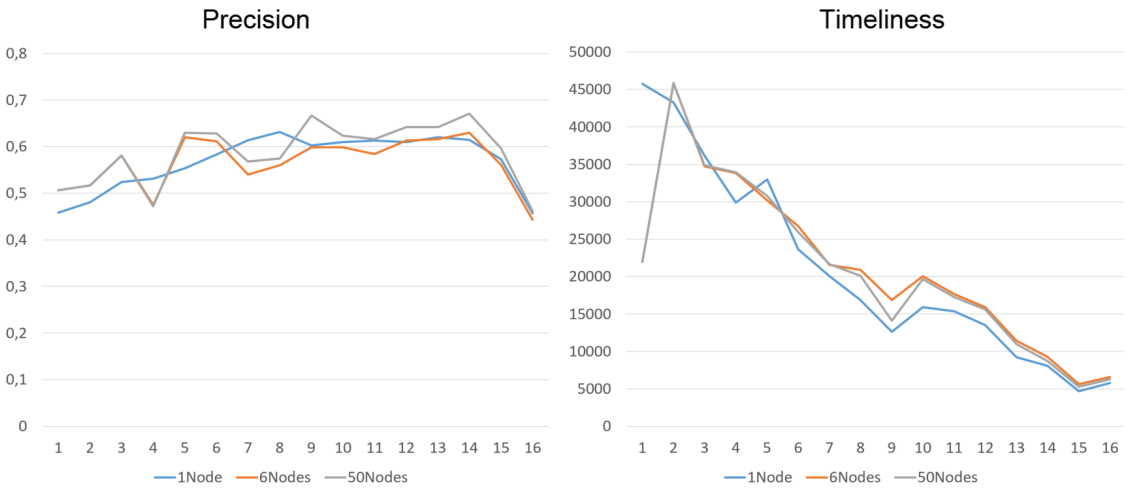

Figure 8.12.: Weight detection through *Artificial Neural Networks* with one hidden layer and different numbers of nodes; left: average precision; right: Timeliness values for averaged $MATD_{cleaned}$ given in minutes.

recommendation step. The preprocessing step is performed at regular intervals and is optimized to take about half a minute at maximum. One of the cost-intensive tasks is to convert the original *xAPI* statements into a proprietary format of the *Smart Learning Recommender* optimized to allow fast online recommendations for the user in the demanded situation. These online recommendations are retrieved on average faster than 25 milliseconds per request for the *Advanced Web Technologies* course and thus can compete with the performance of the learning path algorithm.

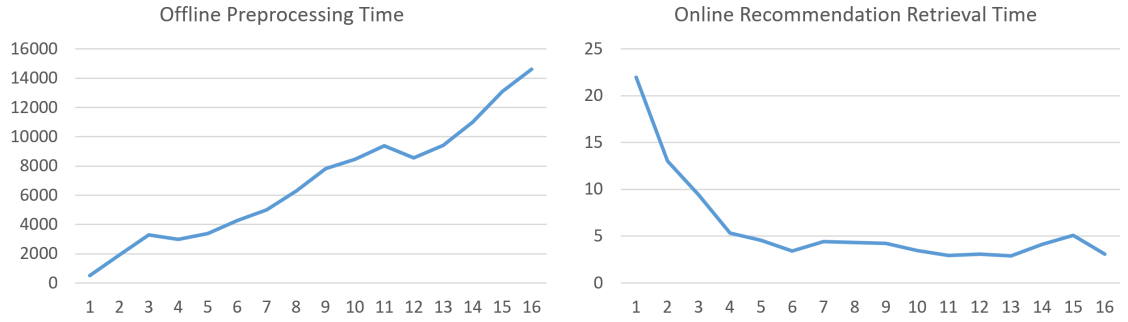

Figure 8.13.: Average calculation time in milliseconds for offline preprocessing (left) and online recommendation retrieval (right).

Comparison of Courses

For a cross course comparison, the *Smart Learning Recommender* algorithm (utilizing the "normalized average factor deviation") has been applied to all three courses: the *Advanced Web Technologies* lecture, the JavaFX online course[141] and the blended learning course of the Energy-Consultant Training[142]. Figure 8.14 visualizes the precision results. As can be seen, the

[141] The *Smart Learning Recommender* applied to the JavaFX course is evaluated according to the evaluation setting in Appendix B.41.
[142] The evaluation setting of the Energy-Consultant Training is presented in Appendix B.42.

incorporation of activities on *Learning Units* results in more precise recommendations compared to those on *Learning Objects*. After a cold-start phase, each of the two courses with face-to-face meetings (the *AWT* and the Energy-Consultant Training) reach almost similar precision values for *LUs* and *LOs*. The precision values of the JavaFX experiments on *LUs* and *LOs*, in turn, are more scattered. This might be due to the fact that only three *Learning Units* are presented which comprise a high number of up to 30 *Learning Objects* per *LU*, while there are much more *LUs* in the other two courses which comprise less *LOs* (10 to 15 on average). At all, the precision values on *Learning Units* are much higher for the Energy-Consultant Training (0.815) and for JavaFX (0.818) than for *AWT* (0.583). This confirms the findings of Verbert et al. [269] that the precision results of educational *Recommender Systems* highly depend on the dataset selection and, thus, implicitly on the the course setting and the course participants.

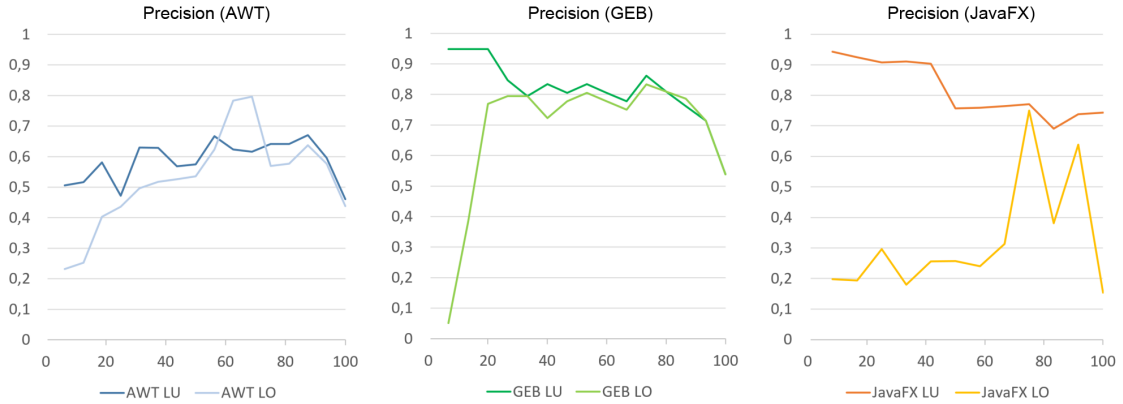

Figure 8.14.: Comparison of the precision for *Learning Units* and *Learning Objects* reached in the three courses: *AWT* (left), Energy-Consultant Training (middle) and JavaFX (right). For the sake of comparability, the time presented on the x-axis is given in percent of the course progress instead of course weeks.

Only Active Students

Similar to the *Collaborative Filtering* analysis, the reasons for the low precision results of the *AWT* course are analyzed for the *SLR* approach. While the *Smart Learning Recommender* also takes repetitions (with the forgetting factor) into account, this strategy seems to heavily rely on a minimum set of user information. Indeed, the lecture factor allows for recommending items without any individually tailored Top-N list. The precision cannot go beyond the lecture precision without any user activity data. Based on the findings of the learning pattern analysis (Section 5.3), only the 26 users of the "Completing" cluster are taken into account in an additional experiment. Finally, another experiment with just the one third (33) most active users in *AWT* has been conducted. Those are the users who interacted at least 106 times with *AWT Learning Units* – remember: the course offered 106 unique *Learning Units* in total.

Figure 8.15 shows an enormous improvement of the precision achieved when only considering particular students. Thereby, more precise recommendations have been calculated using only the

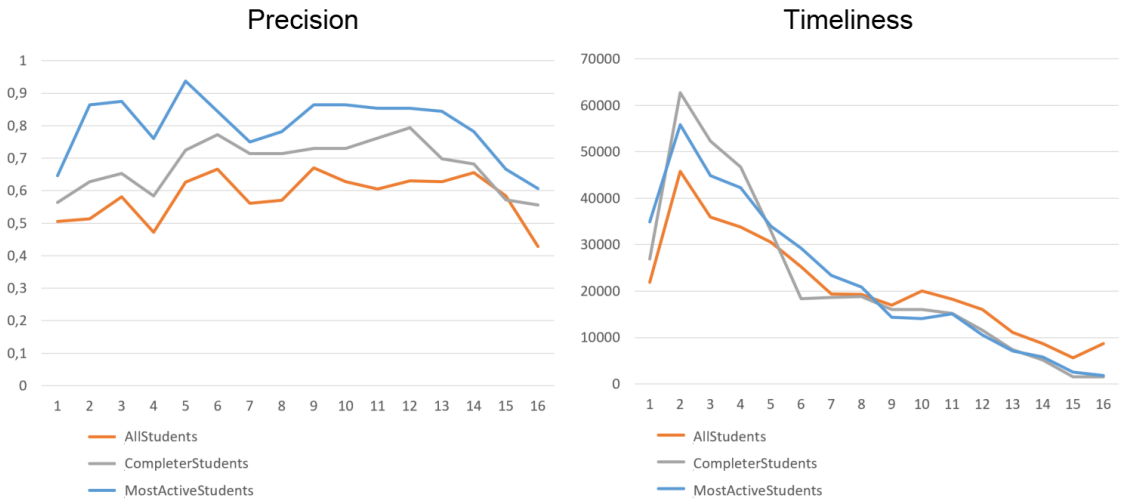

Figure 8.15.: Precision of the optimized *Smart Learning Recommender* approach for all students, only learners of the completer cluster and the most active users.

most active learners, and not – as initially expected – with the most successful learners of the completing cluster. The "normalized average factor deviation" algorithm reaches a precision of 0.800 on average (compared to the previous value of 0.583 with all students). Of course, this is only an artificial setting, because the *SLR* cannot forecast the most active users at an early point in time and, more importantly, the *Recommender System* should provide recommendations for all users – not only for the most active or most successful users. However, this indicates the potential of the algorithm, as the *Smart Learning Recommender* aims at recommending items for all users based on favorable patterns. These patterns might be, for instance, the activity patterns of the most active users.

General Comparison

In the context of all evaluated *Recommender Systems*, the *Smart Learning Recommender* performs best on average regarding precision. Figure 8.16 visualizes the precision and timeliness of all evaluated approaches – each given in the best setting for the Top-3 recommendations for the *AWT* course. As seen on the precision chart, on a weekly basis, the *SLR* (red line) is more scattered than the Learning Path algorithm that shows a worse but more stable trend. Regarding timeliness, the Learning Path algorithm still outperforms the *Smart Learning Recommender*. However, the *Smart Learning Recommender* also shows low timeliness values that are the second best on average.

Algorithm Decision

For practical reasons, the two most promising approaches will be integrated into the *Learning Companion Application* in conjunction[143]: The learning path visualization gives a good overview of already studied items and the thematic opportunities the learner has to explore next. The possibilities are visualized as branches and thus do not need any further explanation. However,

[143]Note that the original *Learning Companion Application* that was used for the data collection only offered recommendations of the *Smart Learning Recommender*.

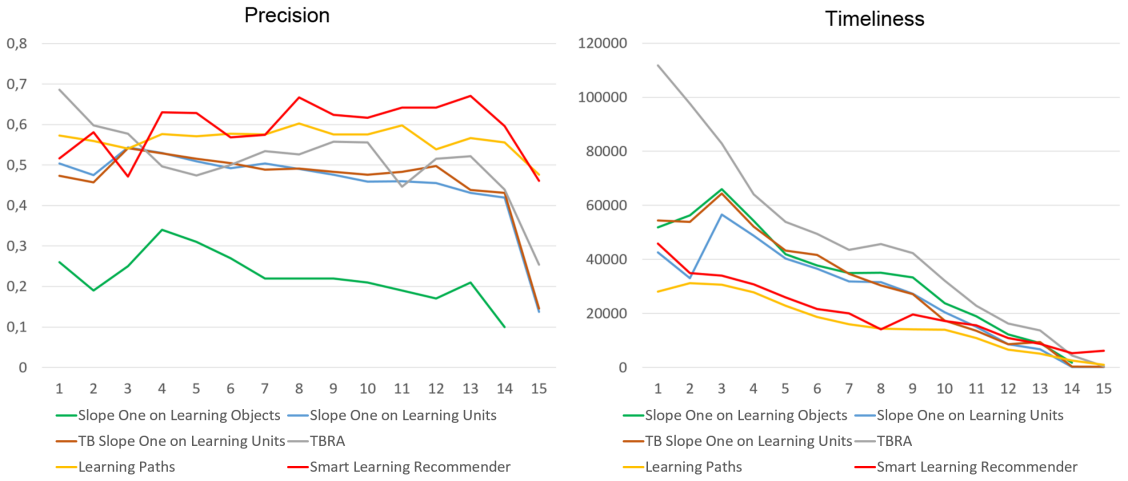

Figure 8.16.: Comparison of the evaluated approaches including the *Smart Learning Recommender* (each given in the best setting); left: average precision; right: Timeliness values for averaged $MATD_{cleaned}$ given in minutes.

the learning path algorithm shows some weaknesses. Especially for *Self-Regulated Learning* when users follow individual approaches to knowledge acquisition and even skip some topics, learning paths are likely to fail or at least recommend omitted items repeatedly. To compensate for these weaknesses, the *LCA* additionally presents the more precise recommendations of the *Smart Learning Recommender*. It not only gives precise hints for learning. It, moreover, allows for detailed insights into the learning progress to be gained in different categories (presented as factors) and, finally, also offers recommendations for those particular categories. With *SLR*, learners can individually decide what aspect they want to focus on next – such as forgotten content, items with weak exercise results or even topics that are part of the next face-to-face-lecture.

Next Chapter

This chapter introduced a distinct approach for recommending course materials through the incorporation of multi-context attributes, called factors, and time weights that credit the time dependency of a course. Based on the findings of the evaluations, the next chapter discusses the presented hypotheses.

9. Discussion of the Hypotheses

About this Chapter

The defined scientific hypotheses follow common formal rules applied to scientific works [28, 265, 44]. This chapter lists all of the previously stated hypotheses (in reverse order of the chapters) and discusses their validity based on the conducted experiments, evaluations and theoretical considerations.

9.1. Evaluation Framework

Scientific Hypothesis /SH5.0/: **An educational *Recommender System* for closed-corpus recommendations may be evaluated by following a specialized evaluation framework in order to produce relevant results. If a traditional evaluation procedure is applied that comprises, e.g., an n-fold cross-validation, its results would not reflect the time-dependent conditions of a closed course. Thus, in order to produce relevant results, a specialized evaluation framework for *Recommender Systems* that aims at predicting course items may utilize a time-dependent cross-validation procedure and may measure the qualitative composition of Top-N lists as well as the timeliness of its recommendations.**

Comparability

Evaluation results are only comparable when applying the same methodological framework to all objects of investigations. This is also reflected by the work of Said and Bellogin, who identified discrepancies in the determined measures due to the following evaluation dimensions: dataset, data splitting, evaluation strategy and metrics [228, p. 129]. Each of the four evaluation dimensions can be seen as a variable, where only one variable is allowed to be changed within a reliable evaluation setting.

When, for example, the analyzed datasets are collected in a different educational context with other context features than the *AWT* dataset, an evaluation must apply the same data splitting approach, the same evaluation strategies (in terms of algorithms) and the same measures in order to produce reliable results. Moreover, the overall recommendation goal of the algorithms must be explained. For instance, how does a forecast of a future item's rating support learners in their learning process? As the *AWT* data do not comprise rating data, it should not be compared to other datasets based on ratings.

External Comparison

The findings of Verbert et al. [269, p. 16] indicate that the measurements of *Recommender*

Systems are highly dependent on the dataset. They measured the F1-score for Top-10 recommendations based on the Tanimoto-Jaccard Coefficient [261] for 4 different datasets (3 *TEL* datasets and the MovieLens dataset). The authors state F1-score values which range, depending on the number of considered users, between about 0.05 up to almost 0.3. In this work, the Top-10 recommendations through the learning paths algorithm reach a F1-score of 0.27 (for the Top-15 even 0.34) and for the optimized *TBRA* approach a F1-score of 0.20 (for Top-30 even 0.39). However, this comparison shows the weaknesses of the common evaluation settings that are presented in the following.

Common Data Basis

The datasets differ in their application area, service origin, quantity and density – which massively impact the measurements, as also noticed by Verbert et al. [269]. Therefore, it is important to evaluate *RSs* based on common data – which leads to a high demand for open educational, academic datasets.

RS Goal

The algorithms and feature selections of Verbert's approach and the analysis in this work differ significantly. However, evaluations should have at least the same goal when they are compared: While Verbert et al. want to prove the appropriateness of the analyzed datasets [269], the evaluation in this work analyzes the appropriateness of different *RS* algorithms for the same dataset. As both evaluations do not have consistency between the evaluation setting regarding algorithm selection and data, the results are not comparable.

Definition of Relevance

While the standard cross-validation setting typically produces reliable results, it does not work within a course setting, where, for instance, every item is relevant (e.g., because all items must be learned to pass the final exam). If every item would be relevant, the cross-validation would produce precision results of 100% every time, because no matter what is recommended, it is automatically relevant. That is why, it is important to carefully define the set of relevant items – in this case, it is the set of all items that have been accessed after the point in time of the recommendation (item accesses in the test set). Thus, results of a traditional evaluation procedure for the analysis of course item recommendations are not comparable with the results introduced in this work.

Data Splitting

The most cross-validation settings are an n-fold cross-validation as known from regular evaluations of *Recommender Systems* [117, 269, 33]. This evaluation is usually time independent. The observation of measures over time, however, is relevant, as at different points in time the algorithms produce varying results. A random split might select data that produce excessively good or bad results for particular approaches. Those temporal effects are disregarded due to the averaging over time.

The validation procedure in this work is an "increasing time-window" cross-validation that better represents real-life conditions because it splits the data according to the time sequence of the collected data [57]. Usually, this manner of splitting gives worse results for precision. However, it better represents the real-world conditions of courses. For instance, the cold start phase is

not considered by a time-independent n-fold splitting where user–item activity is always selected randomly from the whole period of the available data. During the evaluation, different course phases have been determined that influence the precision:

1. Cold Start Phase: At the beginning of a course, the algorithms show low precision as they lack the important user feedback.
2. Guided Learning Period: Within the period of the lectures, when teachers present the actual course materials in face-to-face meetings, or when online educators request assessment submissions at regular intervals, the didactic structure gives a good clue which items are of interest. Algorithms that incorporate the didactic dependencies of items (e.g., the prerequisites or the course schedule – such as the learning path algorithm or the *Smart Learning Recommender*) show higher precision values than the those that rely exclusively on *Collaborative Filtering*.
3. Holidays and Breaks: When there are no face-to-face meetings, learners are required to learn on their own. In this phase, approaches that rely on content metadata and schedules show lower precision values and especially lower timeliness values (as shown by the lecture factor of the *Smart Learning Recommender*).
4. Learning Phase: In courses that end with a final assessment, learners usually increase their learning activities in the last phase before this assessment. These phases produce a massive amount of diverse activity data which result in a reduction of the precision for all evaluated algorithms.
5. After Course Phase: Finally, learners sometimes access the course contents again after the course has ended but with a different goal than to pass the final assessment. Thus, when a course is still accessible after the course ending, the *Recommender System* must treat this phase separately – which is outside of the scope for this work.

The analyzed courses indicate also a high time dependency when analyzing the different course weeks separately. Therefore, a *Recommender System* for course items should be evaluated with a time-dependent evaluation framework.

Error Measures

Error and deviation measurements rely on the numerical range of the feedback data. The predicted relevance scores often use the same format, which might differ even for the same application area. While some approaches try to predict preferences based on ratings (as the original Slope One algorithm does), the introduced educational algorithms aim at predicting the relevance of learning particular items – which is referred to as learning need. Thus, error and deviation measures, such as *RMSE* and *MAE*, are not necessarily comparable for different algorithms. However, precision, recall and F1-score indicate the quality of the Top-N composition independently of the underlying relevance scores. Thus, the precision value is preferred in order to produce comparable results.

F1-Score

Since, especially for settings with a little number of N items in the Top-N list, the recall value is comparatively low, it has a huge effect on the F1-score. This effect is demonstrated with the learning path evaluation. To some extent, the greater the number of items are presented in the Top-N list, the higher the recall value and the higher the resulting F1-score. Thus, analyses of recall values in different Top-N settings would lead to a result that it is better to present more items to the user as higher recall values are reached. However, a very large number of presented recommendations offered at a single glance is counterproductive to the main aim of a *Recommender System*. When the user is overwhelmed by the number of recommendations, the learner's item selection process is not supported at all. In the case of a closed-corpus *Recommender System*, the recall value can be neglected when the learner should learn all relevant items. Thus, it is not important to indicate how many of the relevant items are presented in the Top-N list (recall), but how many presented items are relevant (precision).

Timeliness

Precision, recall and F-measures do not yield information about the timely relevance of the recommendations. Although relevant items are in the test set, they might only become relevant at the end of a course. The introduced timeliness measure, *Mean Absolute Timeliness Deviation*, takes this point into account and indicates an average time deviation which can be stated as an absolute value (in order to classify the results as in the cleaned *MATD*) or as a normalized value presenting the timeliness in relation to the possible time range (as done via the normalized *MATD*). In contrast to precision and recall, the timeliness measure must be as low as possible. However, it works only in conjunction with the precision value, as all non-relevant (or not-accessed) recommendations are not covered by the *MATD* measure. Thus, its main aim is to support the differentiation between algorithms when they show similar precision results.

Limitations

The evaluation framework of Campos et al. [57] also has some drawbacks – especially when the timespan of the dataset is small. For closed-corpus courses, an increasing time-window cross-validation can only be applied in the time between course start and course end. Thereby, the first and the last threshold splits are likely to show worse results compared to the rest. This is an effect of the small dataset sizes: In the case of the conducted courses, the first split (week 1 for training and the rest for testing) does not allow for an adequate training of the algorithms. Moreover, the last split (all but the last week for training and just the last week for testing) does not allow the recommendations to be tested in the same way as done before. The test set comprises a much smaller dataset than the training set and thus it is challenging to identify items for this week. In turn, and as shown, e.g., by the Slope One evaluation, the recall value increases, and in the end indicates the same effect, because there are fewer items in the test set that have a higher probability of being part of the Top-N list. However, the conducted experiments show that the number of activities increases in the final few weeks of a course (especially for courses with final assessments). This lessens the effect at the end. Moreover, the two presented limitations represent

precisely the conditions of a real course where a *Recommender System* does not have very much data at the beginning and where it must recommend very focused items at the end.

Hypothesis

These findings will encourage researchers to follow a special formal evaluation framework which is borrowed from the area of *Time-Aware Recommender System* as it best represents the setting of an educational *Recommender System* for course items. Therefore, researchers must answer the methodological questions of Campos et al. [57, p. 87] and can use common measures if their meaning for the evaluation is well defined. For instance, error or deviation measurements can be applied, but only on the same type of relevance scores. Finally, it might be the case that the timeliness of recommendations is of interest. Then, the proposed *Mean Absolute Timeliness Deviation* represents an appropriate measure to produce comparable and reliable results. Overall, the findings in this work confirm this hypothesis.

9.2. Collected Activity Data

Scientific Hypothesis /SH4.0/: **The activity data collected during the conducted courses, especially the dataset relating to the *Advanced Web Technologies* course, may be utilized to evaluate closed-corpus *Recommender Systems* for the prediction of course items.**

Due to a lack of appropriate academic datasets, the evaluations are based on the collected data in the courses on *AWT*, JavaFX and the Energy-Consultant Training. Since these data have not yet been utilized for other evaluations, their reliability and generalizability might be arguable. According to Drachsler et al. [92], a dataset that is used for evaluating educational *Recommender Systems* should (1) reflect the characteristics of a real-world setting, (2) contain a sufficient number of user profiles and (3) be comparable to other learning settings.

Real-World Characteristics

All three major datasets represent non-synthetic, real-world courses. They directly reflect the learning activities collected through the use of the *Learning Companion Application*. However, at least some learning activities are probably not covered by the datasets, as course participants might also learn offline with external material as well as printed versions of the media presented in the *LCA*. It is hard to measure the amount of offline learning a student engages in, as the system does not receive any such relevant feedback. Most likely, the ratio of off- to on-line learning also depends on the particular course setting. The Energy-Consultant Training, for instance, is held in a Blended Learning setting with presence lectures and a limited number of complementary contents presented in *LCA*. The *LCA* does not collect any information on real presented topics during that time nor on offline learning. The JavaFX course, in turn, is exclusively offered online. As students learn specific programming skills, they have to use external tools for programming which are not monitored for the dataset. And in *AWT*, the same topics are presented in the *LCA*

which are taught during the lectures. Nevertheless, for the course *Advanced Web Technologies*, the *LCA* additionally offered downloadable *PDF* scripts for all 99 registered students. Thereby, between 19 and 66 users downloaded the weekly *PDFs* and, in total, one third of all students downloaded all offered *PDF* files [17, p. 224].

Taking the limitation of not-covered learning steps into account, the collected course data comprise the, in *LCA* gathered, real-world learning activities. Especially for further course iterations, where the setting is almost equal, the data represent realistic user activities.

Number of User Profiles

Depending on the course setting and course runs, the number of users varies between six and eight enrolled participants for the Energy-Consultant Training (all passed the exam), 51 for JavaFX (only 28 used *LCA*) and 126 for *Advanced Web Technologies* (99 used *LCA* and 83 passed the test).

While the number of *LCA* users is quite low (especially in comparison to *MOOCs*), they represent realistic conditions for the educational institutions, Chamber of Crafts, Beuth University and *TU Berlin*. Moreover, the collected activity statements per user are extraordinarily extensive. In JavaFX, 126 *xAPI* statements have been collected on average per student, 273 in the first iteration of the Energy-Consulting course and in *AWT* an average student produced 932 *xAPI* statements. The six analyzed datasets of the dataTEL Challenge [269] show an average number of between 0.92 and 402 activities per user per dataset. Thus, the collected data describe the learning activities in a very detailed way.

Comparability

In the conducted experiments, three different course settings have been analyzed: face-to-face lectures with a digital offering of the lecture topics, Blended Learning with lectures and complementary digital media and an online-only course. The data contain typical learning patterns which are analyzed for JavaFX and in *AWT*. Both show at least a subset of patterns that also other researchers have identified in their courses [155, 109]. With the help of these patterns, the learners can be classified into groups of successful/ "completing" learners, "strong starting" learners or "auditing" learners which are at risk to drop-out when there is a final exam. Moreover, in the *Advanced Web Technologies* course, learners that show these patterns receive average course grades that also reflect their degree of activity [17].

Reliability & Generalizability

The collected data are appropriate to utilize for evaluating educational *Recommender Systems* which are designed for these course settings. The data comprise a qualitative set of *xAPI* statements which cover important aspects of learning activities: e.g., self-assessments, content accesses and exercises. Moreover, the collected data are appropriate for evaluations of different recommender algorithms – which results in comparable measures, as done in this work. Finally, the data show typical learning patterns which have been identified also in other course settings. Thus, the collected datasets are appropriate for evaluations of educational *Recommender Systems* which confirms the hypothesis.

9.3. Learning Environment for Recommender Systems

Scientific Hypothesis /SH3.0/: **An educational *Recommender System* for the recommendation of course items should build on well-selected open specifications for interfaces and data formats. If a course is based on monolithic specifications, such as the *Sharable Content Object Reference Model (SCORM)*, or on proprietary solutions, such as the XBlocks of Open edX, it would impede the utilization of a *Recommender System* for course items. Thus, the atomicity of specifications, such as given in the *Learning Tools Interoperability Specification (LTI)*, the *Learning Resource Meta-data Specification (LOM)*, the *Question and Test Interoperability Specification (QTI)*, the *Common Cartridge Specification (CC)* as well as in the *Experience API (xAPI)*, may be used for the definition of the interfaces and the exchange of the required data for an educational *Recommender System*.**

In recent years, *ADL SCORM* has become popular as a specification for *Technology Enhanced Learning* as it covers all essential aspects: from a standard content metadata description, over content access to the persistence of activity data. However, *SCORM* is often deemed as too inflexible for adaptive learning environments, as course contents are treated as complex, monolithic structures of information that do not allow for the processing of a unique *Learning Object* separately from the entire course structure[144]. This, however, is necessary for adaptive learning technologies and especially for educational *Recommender Systems* which recommend unique items of interest.

Media Presentation

The strength of the *LTI* is that contents can be requested and displayed independently from other contents or specific user interfaces. An *LTI* compliant content server (acting as *LTI* provider) provides a *REST API* for a standardized content access. Content, e.g., an *LO*, has a unique identifier that is represented as a specific *URI* within the *Recommender Engine*, because most algorithms, such as the Slope One, just processes these identifiers as well as the users' activities to generate recommendations.

Item Metadata

While for an educational *Recommender System* the whole set of metadata of an item is not directly of interest – for instance, the *SLR* does not incorporate the given *LOM*, *QTI* or *CC* data – it makes use of some of its features. The hierarchical structure of *Learning Objects*, *Learning Units* and the entire course is stored in a *IMS CC*-compliant way in the repository. The *Recommender System* loads only the required information, e.g., the information of relationships between items, from the repository into a proprietary format given by the *Recommender System*. Other important item-related features are: prerequisites of an item, the typical learning time and the time at an item is presented in the face-to-face phase. All other information, particularly the content itself, is not of interest for the *Recommender System*. Thus, this part of the hypothesis can be disproved,

[144] See: `https://www.efrontlearning.com/blog/2013/04/why-scorm-2004-failed-what-that-means-for-tin-can.html` (Accessed: 07.12.2017).

as the specification of content metadata is, at least for the realized algorithms in this work, not of interest for the *Recommender System*.

XBlocks

The proprietary XBlocks definition of Open edX also represents the course's content metadata which are, as already discussed, not directly of interest for an educational *Recommender System*. However, the original content structure of edX does not allow for the processing and presenting of items separately by a *Recommender System* – which makes it similarly monolithic as *SCORM*. This is why Open edX allows for the integration of external *LTI* tools[145] additionally via an *LTI* consumer plug-in. Moreover, Open edX can also offer its contents via an *LTI* provider[146].

However, if the educational *RS* would build on a similar, but proprietary, interface as with *LTI*, it would not be interoperable with other data sources. Thus, an adapter must be developed for every new content format which would make the development process ineffective at a certain point.

Activity Data

One of the most challenging issues in *Recommender Systems* is the representation of user–item feedback. While the systems often process representations of user–item matrices that are optimized to the algorithms, these models are very static and hardly transferable to other systems. Within the Smart Learning project, different components make use of the same activity data that is stored externally – for instance, a Learning Analytics module or the learner's user interface, to display the last activities, such as wrongly answered questions. That is why the data are stored in a particular way according to the *xAPI* specification. The educational *Recommender Systems* in this work, however, transfer this information at regular intervals into more efficient (proprietary) databases[147]. This transfer is necessary, as the *Experience API* is designed as an all-embracing structure with a lot of unimportant information for *Recommender Systems*[148]. Moreover, a *Learning Record Store*, such as Learning Locker, is designed as a document store such that every document must be accessed and parsed to filter or sort items by specific content values. This makes *xAPI* inappropriate for use in a *RS* and requires the transfer of activity data into proprietary formats.

Reusability

The *SLR*, as well as the other components of the Smart Learning Infrastructure, come as closed components following the paradigm of "high cohesion and low coupling". This means that the *Smart Learning Recommender* contains all important layers that are required to recommend items; an interface to load the item and activity data, a database to store and access the data efficiently, a

[145] LTI in Open edX. See: `https://open.edx.org/learning-tools-interoperability` (Accessed: 07.12.2017).

[146] Open edX as an LTI Tool Provider. See: `https://open.edx.org/blog/open-edx-lti-tool-provider` (Accessed: 07.12.2017).

[147] For instance, the *Smart Learning Recommender* uses MySQL with a proprietary database scheme that is not known by other *Recommender Systems*. The Learning Path algorithm uses a Neo4J database and the Slope One algorithm uses a file-based format of "R". All databases are optimized for the algorithms that process the data but are not transferable to other systems.

[148] An export of the *xAPI* data of the *AWT* course from Learning Locker comprises about 190 MB *JSON* data with about 50 different values per activity – see Appendix B.24 for an example activity statement. The *SLR* requires only 5 to 10 of these values.

service logic layer to predict recommendations and an interface to return recommendations. While the database and the service logic layer are presented as a black box for third-party stakeholders, the interfaces work on the previously mentioned open specifications which makes it interoperable in all environments following the same specifications. Course contents and recommendations can be presented as *LTI* tools in different *Learning Management Systems*. This would not be possible with monolithic specifications, such as *SCORM*, and would result in inflexibility with proprietary formats. However, the appropriateness of other recent specifications, such as *IMS* Caliper, has not been evaluated and thus the utilized specifications represent only one possible approach. Nevertheless, this work demonstrates the strengths, especially the interoperability of components and contents, through the incorporation of open specifications of *Technology Enhanced Learning*. Thereby, *LTI*, as well as *xAPI*, are of particular interest, while *IMS Learning Resource Meta-data Specification*, *Question and Test Interoperability Specification* and *Common Cartridge Specification* do not directly impact the educational *Recommender Systems*. Thus, the hypothesis can be confirmed in general – although the proposed specifications do not represent necessarily the best nor even an exclusive solution to the problem of interoperability.

9.4. The Special Educational Recommendation Paradigm

Scientific Hypothesis /SH2.0/: **An educational *Recommender System* that recommends course items should respect the special paradigm for the recommendation of course materials. The paradigm comprises various aspects, for instance, a learning-oriented relevance score that is called the learning need, the necessity of incorporating multi-dimensional context attributes and the time dependency of the data that impact the precision of the recommendations. If a *Recommender System*, that does not respect, or only partially respects, the special paradigm, is applied for the recommendation of course items, it would generate less precise Top-N recommendations.**

Paradigm

The realized *Recommender System* algorithms are focused on the prediction of items within a course. Thereby, the recommended items should make learning more efficient and effective. Efficiency describes the means of knowledge acquisition and effectiveness describes the quantity and quality of the acquired knowledge.

Efficiency

A preliminary Thinking-Aloud Study with participants of the *AWT* course indicates that learners are similarly efficient in accessing items when utilizing the recommendations compared to the utilization of a search engine, and are more efficient than manually searching for specific *Learning Objects*[149]. While it is hard to measure whether the *Learning Companion Application* with the underlying *Smart Learning Recommender* really shortens the time needed for knowledge

[149] A preliminary Thinking-Aloud Study was conducted where 29 participants of *AWT* performed typical tasks on the *Learning Companion Application*. This study is presented in Appendix B.55.

acquisition, it allows for an exhaustive understanding of learning patterns. Two course runs of the *Advanced Web Technologies* course in two subsequent years were compared: a first without the *Learning Companion Application* and a second that offers the *LCA* (including the *Smart Learning Recommender*). In the course run with the *LCA*, more students were encouraged to participate in the final exam compared to the course run without the *LCA*. This is reflected in the absolute numbers of participants (83 with *LCA* compared to 39 without) as well as by the drop-out ratio (only 34% drop-outs with *LCA* and 46% without).

Effectiveness

The exam results of both *AWT* runs (the run with and the run without the *Learning Companion Application*) were similar, with only a small improvement of the average mark (1.9 with *LCA*; 2.0 without). These numbers are not significant however because only two courses have been compared so far – and the improvement might have occurred for other reasons. However, this means that the *LCA* has at least no negative effect on the efficiency of the students, which is a very good result as the number of participants doubled for the course run with the *Learning Companion Application*. Moreover, the most prominent advancement comes from the ability to analyze the learning patterns and draw conclusions for both learners and educators, and, consequently, to optimize the content offering.

Motivation

While the theoretical user acceptance of a *Recommender System* is measured through the precision and timeliness of the recommendations, a motivational *TEL* recommender must challenge or even provoke the user to make the learning process more effective [277, 191]. Otherwise, the algorithms would just recommend exactly the (appropriate or inappropriate) future activities that the learner would seek anyway. This might work for successful learners whose learning patterns are good enough to successfully pass the assessments (even without a *Recommender System*). Weak learners and those who are at risk of dropping-out however should not receive recommendations based on their inappropriate learning patterns, but on the successful learning patterns that have worked for others.

Successful Patterns

The work in Section 5.3 indicates that learners on a course can be clustered according to typical learning activities. Thereby, the work follows clustering approaches and patterns that have been identified by other researchers [155, 109]. These methods allow for a separation of successful learners from weak or drop-out learners and their collected activity data. For a motivating and compelling *Recommender System*, especially for one that uses *Collaborative Filtering*, only the activities of successful learners should be utilized as a training set. As a result, effective and weak users would both receive recommendations based on the patterns of effective users who achieved good results at the end. This pattern transfer comes with a decrease of the precision according to the evaluation framework, because the analyzed recommendations would not represent an extrapolation of the user's actual activities and so not match the data in the test sets. Thus, the consideration of only successful pattern cannot be evaluated with the defined evaluation framework

– which is a limitation of all evaluation frameworks that are based on historical datasets.

Learning Need

The *Smart Learning Recommender* introduces a relevance score that represents a numerical value indicating the need for learning (implicitly the knowledge gap) of a learner. This learning need is modeled through different factors and the resulting recommendations are presented in particular factor categories. Therefore, another qualitative study was conducted with the participants of the Chamber of Crafts[150]. The eight participants of the first Energy-Consultant Training were asked after end of the course to complete a survey. As one of the most important features that needs to be offered by an *Learning Management System*, seven of eight participants (88%) wanted a *Recommender System* that gives hints for exam-oriented learning as presented by *LCA*. The same number believe that educational recommendations for course items are useful or very useful – and only one answered neutrally. Six out of eight participants (75%) wanted similar overviews of their learning progress to those presented in *LCA* for the future. Figure 9.1 shows the perception of different *SLR* factors.

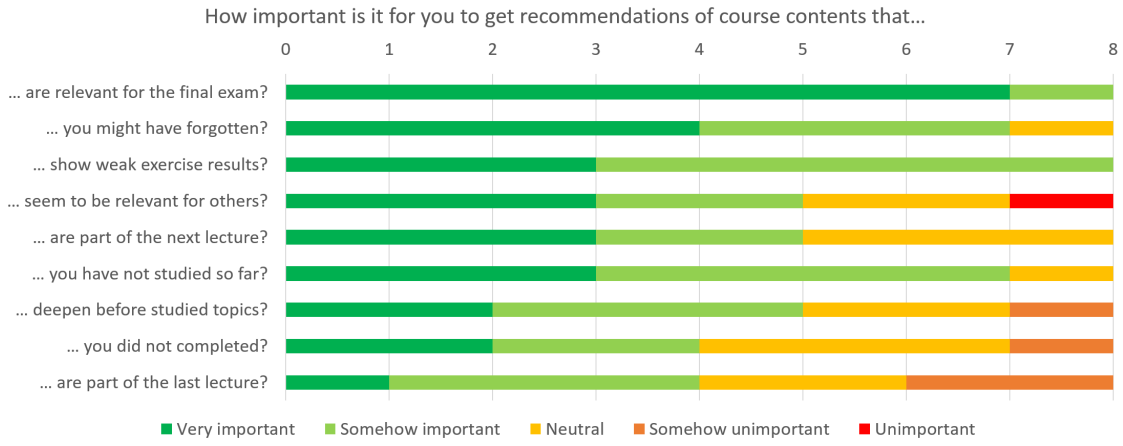

Figure 9.1.: How important are hints for learning contents based on the following factors? (Eight participants of the Chamber of Crafts Berlin were asked; questions provided by Christopher Krauss; survey conducted by Dr. Sarah Hackfort from *IZT*.)

Exam relevance was mentioned as the most important aspect that a *Recommender System* should target on. This was followed by items a learner might have forgotten and *Learning Objects* with a high ratio of wrongly answered questions. In contrast, *Learning Objects* that the user did not completed or contents of the last lecture are of less interest. In general, the participants wanted support in identifying those aspects of the contents that they are not directly aware of, such as exam relevance or forgotten items.

Knowledge-based Filtering

From an algorithmic point of view, the ratios between the context dimensions are not static.

[150] The Smart Learning project partner *IZT* conducted the qualitative survey and included questions of relevance for this dissertation. The study was conducted by Dr. Michael Scharp and Sarah Hackfort. They included the presented questions that have been formulated by Christopher Krauss. More answers of this study are shown in Appendix B.56.

The *SLR* shows that precision can be increased dramatically when modeling context factor weights over time. Thereby, the weights are determined on a weekly interval to understand the best composition. A dynamic weighting at every point in time allows for the generation of the most precise Top-N list[151]. The weight detection was developed to dynamically determine the most appropriate weights based on the most recent activities of the current user.

SLR Paradigm

Table 9.1 compares the aspects of the special paradigm and whether those aspects are taken into account by the evaluated algorithms (for the sake of clarity the algorithms are abbreviated[152]). The Slope One algorithm is borrowed from traditional domains, such as movie recommendations. The *Smart Learning Recommender* follows the defined paradigm and generates more precise recommendations.

Table 9.1.: Differentiation and Evaluation Results of Algorithms

Aspect of Ed-Paradigm	SO	LN-SO	TB-SO	TBLN-SO	TBRA	LP	SLR
Learning Need	-	X	-	X	X	X	X
Repetitions Allowed	X	X	X	X	X	-	X
Warm Start	-	-	-	-	-	X	X
Multi-Attributes	-	-	-	-	-	X	X
Item-Interdependency	-	-	-	-	X	X	X
Time-Dependency	-	-	X	X	X	X	X
Precision	0.459	0.475	0.465	0.485	0.512	0.564	0.587
MATD in min	26,648	32,132	30,107	29,390	36,852	18,340	20,607

In general, the greater the number of aspects of the educational paradigm considered by an algorithm, the more precise are the recommendations. Additionally, the importance of the paradigm is also reflected in the *Mean Absolute Timeliness Deviation* but not in such a clear way. While this work does not allow for a differentiation as to which aspect of the paradigm is more important or has a bigger impact on the precision, it does show that algorithms which do not follow this paradigm yield worse results compared to those that are specialized for *Technology Enhanced Learning*. Which is a confirmation of the hypothesis.

9.5. Traditional RS Techniques for TEL

[151] Due to the evaluation settings, time weights have only been analyzed at weekly intervals.

[152] The Abbreviations in Table 9.1 are defined as follows: SO - Slope One on *Learning Units*; LN-SO - Slope One on *Learning Units* that incorporates the learning need; TB-SO - Slope One on *Learning Units* that incorporates time-weights; TBLN-SO - Slope One on *Learning Units* that incorporates time-weights and the learning need; *TBRA - Time-based Recommender Approach for Lecture Materials*; LP - Learning Paths; *SLR - Smart Learning Recommender*.

Scientific Hypothesis /SH1.0/: ***Recommender Systems* that predict appropriate recommendations for course items may rely on similar techniques as traditional *Recommender Systems*. The basis is user and item metadata and its algorithms aim at predicting relevance scores in order to generate Top-N lists. Educational approaches face similar issues, such as cold start and sparsity, and the algorithms can be classified according to the general *Recommender System* taxonomy.**

User–Item Data

While the relevance score is not, at least in this work, based on preferences and the ratings represent only one possible type of feedback, the algorithms process other user–item data than traditional *Recommender Systems* which are known from the e-commerce and entertainment domain. However, one of the most important feedback types is the user's access to items. This is typically incorporated by other *Recommender Systems*, as well. Moreover, the addition of further dimensions, in terms of time and other multi-context factors, are not *TEL* specific. Actually, the concepts of *Context-Aware Recommender Systems* and *Time-Aware Recommender Systems* have been developed separately from the educational domain, but the evaluations indicate that these approaches also work very well for closed-course recommendations.

Top-N Lists

The learning path generation also indicates that not every algorithm relies on common relevance scores to predict recommendations. While the transition probability of this algorithm results in some kind of relevance score, the Top-N generation does not follow a simple relevance ordering process. Moreover, the presentation and evaluation of a Top-N list for learning paths do not credit the strength of the algorithm – to visualize item relations that result in personalized path recommendations. Moreover, the paths comprise alternative branches. Thus, new types of recommendations, such as the learning paths, eventually require further evaluation measures in the future.

Cold Start & Sparsity

In general, all of the algorithms encountered similar problems to traditional *RS*, but with a different flavor. The cold start problem, for instance, is a big issue for the recommendation of course items, especially when the user registered on the system just before the start of the course. In contrast to e-commerce or entertainment *Recommender Systems*, initial recommendations can rely on the intended didactic structure and thus recommend non-personalized items with a high probability of appropriateness (the lecture factor has a precision of between 0.3 and 0.4 on average during the cold start phase). However, after the initial phase, when all enrolled participants begin to interact with the learning platform, the *Smart Learning Recommender* receives more valuable activity data, and the recommender's user–item matrix becomes dense very fast.

Machine Learning

Machine Learning, such as *Artificial Neural Networks*, can additionally improve the algorithm results and provide even more precise recommendations compared to a linear model, as shown for the *SLR*. Thereby, domain-specific knowledge of the data is crucial. For instance, the naive predication of a relevance score via an *ANN* results in a precision of only 0.354 while the utilization of the *SLR* model with predicted time-dependent weights through an *ANN* results in a precision

of 0.587, which represents the most precise approach in this work.

RS Classes

It turns out that traditional *Collaborative Filtering* approaches, such as the ordinarily proper performing Slope One algorithm, are likely to fail (or at least offer less precise recommendations) when they are transferred without adjustment to the *TEL* domain. The learning path algorithm, instead, aims at combining the strengths of *Content-based Filtering* and *Collaborative Filtering* and produces more precise recommendations that additionally yield better timeliness results compared to the exclusive *Collaborative Filtering*. In addition, the *Smart Learning Recommender* uses a *Hybrid Filtering* approach that is based on features that might be related to the activities of others (as in *Collaborative Filtering*), item metadata (as in *Content-based Filtering*) and especially the user's past activities modeled as knowledge and learning need in the user profile (as known from *Knowledge-based Filtering*). Thereby, the *SLR* produces the best precision results by combining different filtering and *Machine Learning* approaches which are also common in traditional *Recommender Systems*. Thus, this hypothesis also can be confirmed.

10. Conclusion and Future Work

Summary

Within this work, novel approaches for educational *Recommender Systems* are introduced, realized and evaluated. The scientific question of this dissertation was:

> Scientific Question /SQ/:
> **How can a *Recommender System* predict appropriate items in a closed-corpus course environment?**

Special Paradigm

In order to answer this question, a number of hypotheses have been formulated and ultimately confirmed. Related works have been analyzed in the area of adaptive *Technology Enhanced Learning*, resulting in a **specialized paradigm for educational *Recommender Systems*** that needs to be considered when aiming to make learning more effective and more efficient. Traditional *RS* within the e-commerce and entertainment domains try to forecast preferences regarding items. However, this does not work well for adaptive learning systems which aim at supporting learners in reaching the course goals. Therefore, an "appropriate" educational *Recommender System* should rely on the user's learning activities to approximate knowledge levels and gaps which consequently lead to an estimation of the topics requiring study. It turns out that recommending course items is a problem that depends heavily on time – which affects both the evaluation framework and the algorithm design.

Infrastructure & Specifications

To recommend course materials, a **technical reference architecture** has been developed that allows for the connecting of a *Recommender System* and which presents personalized content recommendations in a *Learning Management System*, such as the *Learning Companion Application*. Therefore, **appropriate interfaces and metadata formats** for adaptive *Technology Enhanced Learning* have been identified and tested. From a recommender's perspective, it makes sense to create a proprietary data model that can be processed efficiently by the algorithms. However, it also requires standardized interfaces and data formats to be as interoperable as possible with other components. The learners' activity data are transferred from a *Learning Record Store* as *xAPI* statements into the recommender's database at regular intervals. Moreover, the content items are represented as *LTI URIs* inside and outside of the *Smart Learning Recommender*.

Datasets

This work identifies a lack of academic *TEL* datasets in general, and in particular for those attributes and course settings that are required for time-dependent evaluations. This is why courses are conducted such that they build on the presented architecture and thereby datasets

are collected that represent **typical real-world learning activities**. By the end of 2017, three Blended Learning course runs of the Energy-Consultant training of the Chamber of Crafts in Berlin had been conducted using this infrastructure and the underlying components, among others, with the *Smart Learning Recommender*. Moreover, two courses had also been run at Beuth University: a JavaFX online course and a Computer Science preparatory course. Finally, data from the two course runs of the *TU Berlin* lecture *Advanced Web Technologies* had been collected where one was accompanied by the *Learning Companion Application*. The anonymized activity data of the particular *AWT* iteration with *LCA* were utilized for an exhaustive evaluation of the different recommender algorithms.

Evaluation Framework

As previously mentioned, recommending course items is a time-dependent problem. Thus, these recommendations need to be evaluated within an appropriate framework, because traditional evaluation procedures, such as the common n-fold cross-validation setting, randomly split the item data regardless of any time constraints. However, a course typically comprises different time phases that must be taken into account for the evaluation. A **reliable evaluation framework** is thus proposed for *Time-Aware Recommender Systems* in general, and *RSs* in *Technology Enhanced Learning* in particular. It is adapted from other research areas and incorporates qualitative and quantitative criteria. The goal of presenting "appropriate" recommendations was translated into measurable values, such as precision and recall calculations. Moreover, a new measurement value has been devised to indicate the time accuracy of the recommendations. The *Mean Absolute Timeliness Deviation* presents the average timespan when a recommended relevant item has been accessed after its recommendation.

Algorithms

Following the defined methodology, existing and new ***Time-Aware Recommender System* algorithms for recommending course items** have been designed, realized and evaluated with the aim of making learning more efficient and effective. The evaluation started with a typical *Collaborative Filtering* technique, the Slope One algorithm, and experiments with two extensions, namely the incorporation of time weights (which only slightly increases the accuracy) and the paradigm shift that was realized through the determination of a learning need value. Then, the only other published time-aware *CF* algorithm was applied which had previously been successfully used in the *TEL* context. This algorithm achieved even better results. An entirely different approach has been realized and evaluated with the learning path detection that not only recommends appropriate content items but also visualizes the past activity path and presents new recommendations as branches for future study suggestions. Finally, the *Smart Learning Recommender* was introduced as a context-aware and time-dependent *Knowledge-based Filtering* system that relies on multiple activity and item factors – such as lecture times, self-assessments and item accesses but also on even the learning need of others or the user's predicted memory. Each context factor represents the learning need in studying a particular *Learning Object*. Thereby, the

challenge was to determine appropriate time weights for each factor in the given situation in order to reach the highest possible accuracy. In the end, the *SLR* outperforms the other approaches as it calculates the most precise Top-N lists. However, the learning path algorithm achieves better results regarding the new measurement *MATD* and allows learners to better observe the current course progress. This is why both approaches will be presented in the *LCA* in the future – separately to visualize learning paths and to present factor-specific recommendation categories.

Discussion

At the end of this dissertation, the appropriateness of the algorithms and the key outcomes of this work have been discussed which represents a good starting point for other researchers and developers in the future and can act as a **decision support mechanism for the selection of educational recommender techniques**.

10.1. Limitations of this Work

Acceptance

Most of the evaluations have been conducted in a simulation environment – with historical real-world data but in the absence of presenting the recommendations of all evaluated algorithms to real-world learners. Thus, the effect of the user interface and its usability have not been compared for the different approaches. Since these aspects have a huge impact on general user acceptance and user perception of the system as a whole, they require further experiments.

Limited Data

While the different algorithms were evaluated with the help of a novel evaluation framework, the results can only indicate their various levels of appropriateness for the self-collected datasets. In general, the approaches should work similarly well on other datasets that offer the same kind of user, item and activity information. However, due to the lack of published appropriate datasets, this has not been as yet undertaken. Thus, this work will encourage other researchers or educational staff having the same kind of data to apply and test the presented and additional *Recommender Systems* with the introduced evaluation framework.

Legal Aspects

Greller and Drachsler [116] note that the key constraints of educational *Recommender Systems* can be classified as "legal protection", "privacy", "ethics" and "ownership". However, those are outside of the scope of this dissertation. While these are important aspects of the application in established services, it represents a huge additional field of research with a lot of publications and laws. Moreover, at the time of writing, some existing German laws (e.g., *German Bundesdatenschutzgesetz (BDSG)*, *German Telemediengesetz (TMG)*) are going to be replaced by the EU through the legal validity of the *General Data Protection Regulation (GDPR)*, the as yet still discussed ePrivacy Directive and even the early version of the *Audiovisual Media Services Directive (AVMSD)*. At the very least the *GDPR* will be effective from May 2018 and establishes two new key paradigms: "Privacy by Design" and "Privacy by Default", where all users need to actively agree (via opt-in) before user data can be collected and processed. As the *Learning Companion Application* requires a registration anyway, this confirmation should not be a problem.

However, the legal implications must be discussed with all involved stakeholders.

Privacy Recommendations

In parallel, researchers have started to present policies [92, 136] and guidelines for privacy-compliant data processing, such as "Privacy as Confidentiality", "Privacy as Control" and "Privacy as Practice Paradigm" [270, p. 331]. Moreover, Duval even lists that "the data about a person's attention remain the property of that person", "it should be possible to move data about a person out of one system and into another system", "a person should be able to sell data about his attention" and "it should always be clear to a person that she is being tracked" [97, p. 7].

Ethical Considerations

There are also ethical implications of this work. The *Smart Learning Recommender* is not expected to replace teachers or other educational staff, but rather to assist learners in their knowledge acquisition. However, students might blame these algorithms in future if they fail at their exams or do not reach the expected course goals. Even the most precise *Recommender Systems* (e.g., known from Netflix) can only reach accuracy values of approximately 80% [153] although these accuracy results represent just a statistical probability for all users and some users will receive even less precise recommendations. Even worse, learning patterns which worked for one student will not necessarily work for another. Presenting recommendations means to change future behaviors [144]. In addition, some researchers argue that "unbiased computational processes can lead to discriminative decision procedures" [55] and, thus, should be integrated with care [30]. These aspects are not part of this work and must be analyzed in the future.

10.2. Future Work

During this work, a lot of exciting areas have been identified that require further research. Besides the previously mentioned critical evaluations that would make the results generalizable and representative, there are a number of other open issues.

Extension of LCA

In a further research project, the *Learning Companion Application* is extended via a number of different features. The learning paths that are still calculated offline at the moment will also be presented to the learner. The *SLR* will provide recommendations via the *LTI API* in order to make them reusable in other *Learning Management Systems*. Moreover, users will not only receive item recommendations when working with the *LCA*, but also push notifications and e-mails at regular intervals and in situations that seem to be appropriate for learning.

Further UI Extensions

Other extensions are part of submitted project proposals: For instance for a chatbot that can be integrated in the *LCA* and in external messenger applications (such as Slack[154] or Telegram[155]) that give textual hints in appropriate situations. Moreover, this digital chat avatar should also respond to user questions regarding the course, present current knowledge levels and/or present

[153] The team "BellKor's Pragmatic Chaos" won the Netflix Prize with a *RMSE* of 0.8567 on a 1 to 5 stars scale [158, 217, 266].
[154] Slack. See `https://slack.com` (Accessed: 03.11.2017)
[155] Telegram. See `https://telegram.org/` (Accessed: 03.11.2017).

simple contents and definitions (e.g., from the glossary). Moreover, Gamification elements are planned that will improve the learner's motivation and at the same time enable the collection of more valuable data for the *Recommender System*. Some researchers have even evaluated the effect of observed emotions on learning progress [111] which is an interesting future direction.

Advanced Learning Analytics

The understanding of typical learning patterns, how and when these patterns can be identified during a course, helps to improve the *Recommender Systems* as recommendations are then not only based on the predicted future behavior of the concerned learners but also on successful activity patterns which should bring further benefits, even to weak students.

Global Data Infrastructures

The content creation of digital media is a very cost-intensive process. Thus, it makes sense to share these contents with other organizations, establishing synergies and more efficiently optimizing of the contents due to a greater audience reach. There are some initiatives, such as "Towards Global Data Infrastructures" which represent a collection of all available (learning) materials [270], or the *Open Educational Resources (OER)* [81, 82, 143]. The contents of the Smart Learning project can be shared between organizations similarly, but requires some work on a license model with the capability to protect content and limit specific institutions, regions or times.

Transfer to other domains

A patent has been submitted (see Appendix A.2) that comprises elements of this work, such as the time-dependent, context-aware analysis of various factors that is similar to the presented *Smart Learning Recommender*. Its evaluation procedure follows the framework definition introduced here. It would be interesting to see how other application areas benefit from the findings in this work.

10.3. Closing Word

In recent years, I observed a shift in the focus of the educational industry. While some years ago the most popular term was "mobile learning", it then became "adaptive learning". However, the meaning of adaptive learning still seems to be a moving target. Some service providers implement only simple features for content filtering and call this an adaptive system; others follow more sophisticated approaches by individualizing offerings based on personalized algorithmic forecasts. With this work, a promising direction has been demonstrated combining two fields: *Recommender Systems* and *Technology Enhanced Learning*. The evaluation results demonstrate the feasibility of the system, and the reactions of the industry confirm the relevance of this topic. Thus, I forecast: by 2030 at the latest, learners will not be able to imagine how knowledge acquisition could ever have taken place without the personalized adaption of software to the individual learning needs.

Bibliography

[1] LOM - IEEE Learning Resource Meta-Data Specification. Technical report, IMS Global Learning Consortium, https://www.imsglobal.org/metadata/index.html, 2006.

[2] CP - Content Packaging Specification. Specification, IMS Global Learning Consortium, https://www.imsglobal.org/content/packaging/index.html, 2009.

[3] CC - Common Cartridge Specification. Specification, IMS Global Learning Consortium, http://www.imsglobal.org/cc/index.html, 2015.

[4] LRMI Metadata Terms in RDF. Specification, LRMI - Learning Resource Metadata Initiative, http://dublincore.org/dcx/lrmi-terms/, 2015.

[5] LTI - Learning Tools Interoperability Specification. Specification, IMS Global Learning Consortium, https://www.imsglobal.org/activity/learning-tools-interoperability, 2015.

[6] Qti - question and test interoperability specification. Specification, IMS Global Learning Consortium, https://www.imsglobal.org/question/index.html, 2015.

[7] P. Adibi and B. T. Ladani. A collaborative filtering recommender system based on user's time pattern activity. In *5^{th} Conference on Information and Knowledge Technology (IKT), 2013*, pages 252–257. IEEE, 2013.

[8] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems (TOIS)*, 23(1):103–145, 2005.

[9] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[10] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In *Recommender systems handbook*, pages 217–253. Springer, 2011.

[11] C. C. Aggarwal and C. K. Reddy. *Data Clustering: Algorithms and Applications*. CRC Press, 2013.

[12] Á. F. Agudo-Peregrina, Á. Hernández-García, and S. Iglesias-Pradas. Predicting academic performance with learning analytics in virtual learning environments: A comparative study of three interaction classifications. In *International Symposium on Computers in Education (SIIE)*, pages 1–6. IEEE, 2012.

[13] H. J. Ahn. A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. *Information Sciences, Elsevier*, 178(1):37–51, 2008.

[14] E. Aimeur, G. Brassard, J. M. Fernandez, and F. Onana. Privacy-preserving demographic filtering. In *Proceedings of the 2006 ACM symposium on Applied computing*, pages 872–878. ACM, 2006.

[15] A. Alammary, J. Sheard, and A. Carbone. Blended learning in higher education: Three different design approaches. *Australasian Journal of Educational Technology*, 30(4), 2014.

[16] L. A. Alvarez-Gonzalez, A. Campos, M. E. De la Maza, and D. Ojeda. Interactive assessment learning environment system under ims-qti specification. In *2014 The International Conference on Education Technologies and Computers (ICETC)*, pages 7–11. IEEE, 2014.

[17] T.-S. An, C. Krauss, and A. Merceron. Can typical behaviors identified in moocs be discovered in other courses? In *Proceedings of the 10th International Conference on Educational Data Mining (EDM 2017)*, 2017.

[18] E. Anagnostopoulou, I. Mavroidis, Y. Giossos, and M. Koutsouba. Student satisfaction in the context of a postgraduate programme of the hellenic open university. *Turkish Online Journal of Distance Education, TOJDE, ISSN 1302-6488*, 16(2), 2015.

[19] T. Anderson. *The theory and practice of online learning.* Athabasca University Press, ISBN: 0-919737-59-5, 2008.

[20] J. Aronson, J. Zimmerman, and L. Carlos. Improving student achievement by extending school: Is it just a matter of time?. *ERIC Number: ED435127, PACE Media/Education Writers Seminar*, 1999.

[21] Y. Atif, R. Benlamri, and J. Berri. Learning objects based framework for self-adaptive learning. *Education and Information Technologies*, 8(4):345–368, 2003.

[22] H. Avancini and U. Straccia. User recommendation for collaborative and personalised digital archives. *International Journal of Web Based Communities*, 1(2):163–175, 2005.

[23] L. Averell and A. Heathcote. The form of the forgetting curve and the fate of memories. *Journal of Mathematical Psychology*, 55(1):25–35, 2011.

[24] H. P. Bahrick and E. Phelps. Retention of spanish vocabulary over 8 years. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 13:344–349, 1987.

[25] L. Baltrunas and X. Amatriain. Towards time-dependant recommendation based on implicit feedback. In *Workshop on context-aware recommender systems (CARS09)*, 2009.

[26] L. Baltrunas and F. Ricci. Context-based splitting of item ratings in collaborative filtering. In *Proceedings of the third ACM conference on Recommender systems*, pages 245–248. ACM, 2009.

[27] H. Balzert. *Lehrbuch der Softwaretechnik – Basiskonzepte und Requirements Engineering (3. Aufl.)*. Spektrum - Akademischer Verlag, 2009.

[28] H. Balzert, M. Schröder, and C. Schäfer. *Wissenschaftliches Arbeiten: Ethik, Inhalt & Form wiss. Arbeiten, Handwerkszeug, Quellen, Projektmanagement, Präsentation*. W3L-Verlag, 2011.

[29] K. Bauman and A. Tuzhilin. Recommending remedial learning materials to the students by filling their knowledge gaps. In *Proceedings of the international Conference on Educational Recommender Systems (EdRecSys16)*, 2016.

[30] G. S. Becker. *The economics of discrimination*. University of Chicago press, 2010.

[31] G. Beham, H. Stern, and S. Lindstaedt. Aposdle-ds a dataset from the aposdle workintegrated learning system. In *1st Workshop on Recommender Systems for Technology Enhanced Learning (RecSysTEL 2010)*, 2010.

[32] S. M. Beitzel, E. C. Jensen, A. Chowdhury, O. Frieder, and D. Grossman. Temporal analysis of a very large topically categorized web query log. *Journal of the Association for Information Science and Technology*, 58(2):166–178, 2007.

[33] A. Bellogin, P. Castells, and I. Cantador. Precision-oriented evaluation of recommender systems: an algorithmic comparison. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 333–336. ACM, 2011.

[34] J. Bennett and S. Lanning. The netflix prize. In *Proceedings of KDD cup and workshop, San Jose, California*, volume 2007, page 35, 2007.

[35] T. Berka and M. Plößnig. Designing recommender systems for tourism. ENTER 2004: 11th International Conference on Information Technology in Travel & Tourism, Cairo, Egypt, 26?28 January 2004, 2004.

[36] J. Berri, R. Benlamri, and Y. Atif. Ontology-based framework for context-aware mobile learning. In *Proceedings of the 2006 international conference on Wireless communications and mobile computing*, pages 1307–1310. ACM, 2006.

[37] M. Berry and G. Linoff. *Mastering data mining: The art and science of customer relationship management*. John Wiley & Sons, Inc., 1999.

[38] M. J. Berry and G. Linoff. *Data mining techniques: For marketing, sales, and customer support*. John Wiley & Sons, Inc., 1997.

[39] R. Bessa Babo, A. I. Azevedo, and J. Suhonen. Students' perceptions about assessment using an e-learning platform. In *2015 IEEE 15th International Conference on Advanced Learning Technologies (ICALT)*, pages 244–246. IEEE, 2015.

[40] B. S. Bloom, C. of College, and U. Examiners. *Taxonomy of educational objectives*, volume 2. Longmans, Green New York, 1964.

[41] O. Bohl, J. Scheuhase, R. Sengler, and U. Winand. The sharable content object reference model (scorm)-a critical review. In *Proceedings of the International Conference on Computers in Education*, pages 950–951. IEEE, 2002.

[42] H. Boley. Racofi: A rule-applying collaborative filtering system. in 2003 ieee. In *WIC International Conference on Web Intelligence/Intelligent Agent Technology*, pages 430–434, 2003.

[43] F. Bonchi. Influence propagation in social networks: A data mining perspective. *IEEE Intelligent Informatics Bulletin*, 12(1):8–16, 2011.

[44] J. Bortz and N. Döring. *Forschungsmethoden und evaluation*. Springer-Verlag, 2013.

[45] M. Brocco, G. Groh, and C. Kern. On the influence of social factors on team recommendations. In *2010 IEEE 26th International Conference on Data Engineering Workshops (ICDEW)*, pages 270–277. IEEE, 2010.

[46] J. Brown. Some tests of the decay theory of immediate memory. *Quarterly Journal of Experimental Psychology*, 10(1):12–21, 1958.

[47] P. J. Brown, J. D. Bovey, and X. Chen. Context-aware applications: from the laboratory to the marketplace. *IEEE personal communications*, 4(5):58–64, 1997.

[48] P. Brusilovsky and N. Henze. Open corpus adaptive educational hypermedia. In *The Adaptive Web*, pages 671–696. Springer, 2007.

[49] P. Brusilovsky and W. Nejdl. Adaptive hypermedia and adaptive web. practical handbook of internet computing. *Baton Rouge*, 2004.

[50] J. Buder and C. Schwind. Learning with personalized recommender systems: A psychological view. *Computers in Human Behavior*, 28(1):207–216, 2012.

[51] R. Burke. Knowledge-based recommender systems. *Encyclopedia of library and information science*, 69(Supplement 32):180, 2000.

[52] R. Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.

[53] R. Burke. Hybrid web recommender systems. In *The adaptive web*, pages 377–408. Springer, 2007.

[54] R. Burke and F. Eskandanian. Collaborative recommendation of informal learning experiences. In *IEEE/WIC/ACM International Conference on Web Intelligence Workshops (WIW)*, pages 66–67. IEEE, 2016.

[55] T. Calders and I. Žliobaitė. Why unbiased computational processes can lead to discriminative decision procedures. In *Discrimination and Privacy in the Information Society*, pages 43–57. Springer, 2013.

[56] P. G. Campos, A. Bellogín, F. Díez, and J. E. Chavarriaga. Simple time-biased knn-based recommendations. In *Proceedings of the Workshop on Context-Aware Movie Recommendation*, pages 20–23. ACM, 2010.

[57] P. G. Campos, F. Díez, and I. Cantador. Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. *User Modeling and User-Adapted Interaction*, 24(1-2):67–119, 2014.

[58] M. A. Chatti, S. Dakova, H. Thus, and U. Schroeder. Tag-based collaborative filtering recommendation in personal learning environments. *IEEE Transactions on Learning Technologies*, 6(4):337–349, 2013.

[59] M. A. Chatti, M. Jarke, and M. Specht. The 3p learning model. *Educational Technology & Society*, 13(4):74–85, 2010.

[60] M. A. Chatti, P. Toreini, H. Thues, and U. Schroeder. Sna-based recommendation in professional learning environments. *Proceedings of The Eighth International Conference on Mobile, Hybrid, and On-line Learning (eLmL 2016)*, 2016.

[61] C.-M. Chen. Intelligent web-based learning system with personalized learning path guidance. *Computers & Education*, 51(2):787–814, 2008.

[62] C.-M. Chen. Ontology-based concept map for planning a personalised learning path. *British Journal of Educational Technology*, 40(6):1028–1058, 2009.

[63] W. Chen, I. Paik, J. Wang, B. T. Kumara, and T. Tanaka. Awareness of social influence on linked social service. In *2013 IEEE International Conference on Cybernetics (CYBCONF)*, pages 32–39. IEEE, 2013.

[64] Y.-n. Chen and M. Yu. A hybrid collaborative filtering algorithm based on user-item. *IEEE Computer Society*, pages 618–621, 2010.

[65] S. Chien and N. Immorlica. Semantic similarity between search engine queries using temporal correlation. In *Proceedings of the 14th international conference on World Wide Web*, pages 2–11. ACM, 2005.

[66] P.-A. Chirita, W. Nejdl, and C. Zamfir. Preventing shilling attacks in online recommender systems. In *Proceedings of the 7th annual ACM international workshop on Web information and data management*, pages 67–74. ACM, 2005.

[67] C. L. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, and I. MacKinnon. Novelty and diversity in information retrieval evaluation. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 659–666. ACM, 2008.

[68] M. T. Cole, D. J. Shelley, and L. B. Swartz. Online instruction, e-learning, and student satisfaction: A three year study. *The International Review of Research in Open and Distributed Learning*, 15(6), 2014.

[69] P. Cortez and A. M. G. Silva. Using data mining to predict secondary school student performance. *Proceedings of 5th FUture BUsiness TEChnology Conference (FUBUTEC 2008)*, 2008.

[70] D. Cosley, S. K. Lam, I. Albert, J. A. Konstan, and J. Riedl. Is seeing believing?: how recommender system interfaces affect users' opinions. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 585–592. ACM, 2003.

[71] P. Cremonesi, F. Garzotto, and M. Quadrana. Evaluating top-n recommendations when the best are gone. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 339–342. ACM, 2013.

[72] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 39–46. ACM, 2010.

[73] Y. Cui and S. Bull. Context and learner modelling for the mobile foreign language learner. *Elsevier System*, 33(2):353–367, 2005.

[74] N. Dabbagh and A. Kitsantas. Personal learning environments, social media, and self-regulated learning: A natural formula for connecting formal and informal learning. *The Internet and higher education, Elsevier*, 15(1):3–8, 2012.

[75] N. Dali Betzalel, B. Shapira, and L. Rokach. Please, not now!: A model for timing recommendations. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 297–300. ACM, 2015.

[76] V. Damnjanovic, S. Jednak, and I. Mijatovic. Factors affecting the effectiveness and use of moodle: students' perception. *Interactive learning environments, Taylor & Francis*, 23(4):496–514, 2015.

[77] J. Daniel. Making sense of moocs: Musings in a maze of myth, paradox and possibility. *Journal of interactive Media in education, Ubiquity Press*, 2012(3), 2012.

[78] A. A. Darabi, E. G. Sikorski, and R. B. Harvey. Validated competencies for distance teaching. *Distance Education, Taylor & Francis*, 27(1):105–122, 2006.

[79] A. Das, S. Gollapudi, R. Panigrahy, and M. Salek. Debiasing social wisdom. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 500–508. ACM, 2013.

[80] A. Dattolo, F. Ferrara, and C. Tasso. The role of tags for recommendation: a survey. pages 548–555, 2010.

[81] C. V. de Carvalho, P. Escudeiro, M. C. Rodriguez, and M. L. Nistal. Sustainability strategies for open educational resources and repositories. In *Latin American Conference on Learning Objects and Technology (LACLO)*, pages 1–6. IEEE, 2016.

[82] B. de los Arcos, R. Farrow, R. Pitt, M. Weller, and P. McAndrew. Personalising learning through adaptation: Evidence from a global survey of k-12 teachers? perceptions of their use of open educational resources. *Journal of Online Learning Research, Association for the Advancement of Computing in Education (AACE)*, 2(1):23–40, 2016.

[83] I. De Waard, A. Koutropoulos, R. J. Hogue, S. C. Abajian, N. Ö. Keskin, C. O. Rodriguez, and M. S. Gallagher. Merging mooc and mlearning for increased learner interactions. *International Journal of Mobile and Blended Learning (IJMBL)*, 4(4):34–46, 2012.

[84] F. H. del Olmo and E. Gaudioso. Evaluation of recommender systems: A new approach. *Expert Systems with Applications, Elsevier*, 35(3):790–804, 2008.

[85] P. Desai and M. Vijayalakshmi. Flipped classroom: An efficient pedagogical tool to teach a course for final year computer science and engineering graduate students. *Journal of Engineering Education Transformations*, pages 306–310, 2015.

[86] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22(1):143–177, 2004.

[87] A. K. Dey. Understanding and using context. *Personal and ubiquitous computing*, 5(1):4–7, 2001.

[88] A. K. Dey, G. D. Abowd, and D. Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-computer interaction, L. Erlbaum Associates Inc.*, 16(2):97–166, 2001.

[89] Y. Ding and X. Li. Time weight collaborative filtering. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 485–492. ACM, 2005.

[90] C. Dongtao and X. Dehua. A collaborative filtering recommendation based on user profile weight and time weight. In *International Conference on Computational Intelligence and Software Engineering (CiSE 2009)*, pages 1–4. IEEE, 2009.

[91] M. Dorigo and L. M. Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on evolutionary computation*, 1(1):53–66, 1997.

[92] H. Drachsler, T. Bogers, R. Vuorikari, K. Verbert, E. Duval, N. Manouselis, G. Beham, S. Lindstaedt, H. Stern, M. Friedrich, et al. Issues and considerations regarding sharable data sets for recommender systems in technology enhanced learning. *Procedia Computer Science, Elsevier*, 1(2):2849–2858, 2010.

[93] H. Drachsler, H. Hummel, B. Van den Berg, J. Eshuis, W. Waterink, R. Nadolski, A. Berlanga, N. Boers, and R. Koper. Effects of the isis recommender system for navigation support in self-organised learning networks. *Journal of Educational Technology & Society, JSTOR*, 12(3):115–126, 2009.

[94] H. Drachsler, H. G. Hummel, and R. Koper. Identifying the goal, user model and conditions of recommender systems for formal and informal learning. *Journal of Digital Information*, 10(2), 2009.

[95] H. Drachsler, D. Pecceu, T. Arts, E. Hutten, L. Rutledge, P. Van Rosmalen, H. Hummel, and R. Koper. Remashed–recommendations for mash-up personal learning environments. In *Learning in the synergy of multiple disciplines*, pages 788–793. Springer, 2009.

[96] J. Dron, R. Mitchell, P. Siviter, and C. Boyne. Cofind – an experiment in n-dimensional collaborative filtering. *Journal of Network and Computer Applications, Elsevier*, 23(2):131–142, 2000.

[97] E. Duval. Attention please!: learning analytics for visualization and recommendation. In *Proceedings of the 1st international conference on learning analytics and knowledge*, pages 9–17. ACM, 2011.

[98] H. Ebbinghaus. *Ueber das Gedaechtnis: untersuchungen zur experimentellen psychologie.* Duncker & Humblot, 1885.

[99] H. Ebbinghaus. Memory: A contribution to experimental psychology, translated by henry a ruger and clara e bussenius. New York, NY, US: Teachers College Press, 1913.

[100] M. M. El-Bishouty, H. Ogata, and Y. Yano. Perkam: Personalized knowledge awareness map for computer supported ubiquitous learning. *Journal of Educational Technology & Society, JSTOR*, 10(3), 2007.

[101] A. Elbadrawy and G. Karypis. Domain-aware grade prediction and top-n course recommendation. In *ACM Conference on Recommender Systems (RecSys)*, pages 183–190, 2016.

[102] O. H. Embarak. A method for solving the cold start problem in recommendation systems. In *2011 International Conference on Innovations in Information Technology (IIT)*, pages 238–243. IEEE, 2011.

[103] M. Erdt, A. Fernandez, and C. Rensing. Evaluating recommender systems for technology enhanced learning: a quantitative survey. *IEEE Transactions on Learning Technologies*, 8(4):326–344, 2015.

[104] V. Ermolayev, R. Akerkar, V. Terziyan, and M. Cochez. Towards evolving knowledge ecosystems for big data understanding. *Big data computing, CRC Press*, pages 3–55, 2013.

[105] S. E. Fancsali, S. Ritter, M. Yudelson, M. Sandbothe, and S. R. Berman. Implementation factors and outcomes for intelligent tutoring systems: A case study of time and efficiency with cognitive tutor algebra. *Proceedings of 29th International FLAIRS Conference of the The Florida Artificial Intelligence Research Society, AAAI Publications, Key Largo, Florida, USA*, 2016.

[106] S. Fazeli, H. Drachsler, F. Brouns, and P. Sloep. Towards a social trust-aware recommender for teachers. In *Recommender systems for technology enhanced learning*, pages 177–194. Springer, 2014.

[107] R. M. Felder, L. K. Silverman, et al. Learning and teaching styles in engineering education. *Engineering education*, 78(7):674–681, 1988.

[108] R. M. Felder and J. Spurlin. Applications, reliability and validity of the index of learning styles. *International journal of engineering education, TEMPUS Publication*, 21(1):103–112, 2005.

[109] R. Ferguson and D. Clow. Consistent commitment: patterns of engagement across time in massive open online courses (moocs). *Journal of Learning Analytics, UTS ePress*, 2(3):55–80, 2016.

[110] J. Fiaidhi. Recosearch: a model for collaboratively filtering java learning objects. *International Journal of Instructional Technology and Distance Learning*, 1(7):35–50, 2004.

[111] R. Ghali1, C. Frasson, and S. Ouellet. Towards real time detection of learners' need of help in serious games. *Proceedings of 29th International FLAIRS Conference of the The Florida Artificial Intelligence Research Society, AAAI, Key Largo, Florida, USA*, 2016.

[112] K. Gilhooly. Making e-learning effective. *Computerworld*, 35(29):52–53, 2001.

[113] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM - Special issue on information filtering*, 35(12):61–70, 1992.

[114] M. Gorgoglione and U. Panniello. Including context in a transactional recommender system using a pre-filtering approach: two real e-commerce applications. In *International Conference on Advanced Information Networking and Applications Workshops (WAINA'09)*, pages 667–672. IEEE, 2009.

[115] K. Green. Campus it officers affirm the instructional integration of it as their top priority, offer mixed reviews on it effectiveness and outsourcing for online education. campus computing. Encino, CA: Campus Computing Project. Accessed, 2013.

[116] W. Greller and H. Drachsler. Turning learning into numbers–a learning analytics framework. *International Journal of Educational Technology & Society*, 2012.

[117] A. Gunawardana and G. Shani. A survey of accuracy evaluation metrics of recommendation tasks. *Journal of Machine Learning Research, JMLR*, 10(Dec):2935–2962, 2009.

[118] J. Han. Learning fuzzy association rules and associative classification rules. *IEEE - International Conference on Fuzzy Systems*, pages 1454–1459, 2006.

[119] J. Han, M. Kamber, and J. Pei. *Data mining: concepts and techniques*. Morgan kaufmann, 2006.

[120] F. M. Harper and J. A. Konstan. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5(4):19, 2016.

[121] A. C. Harvey. *Forecasting, structural time series models and the Kalman filter*. Cambridge university press, 1990.

[122] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Science+Business Media, 2001.

[123] L. He and F. Wu. A time-context-based collaborative filtering algorithm. In *IEEE International Conference on Granular Computing (GRC'09)*, pages 209–213. IEEE, 2009.

[124] J. L. Herlocker, J. A. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pages 241–250. ACM, 2000.

[125] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.

[126] C. Hermann. Time-based recommendations for lecture materials. In *2010 World Conference on Educational Multimedia, Hypermedia and Telecommunications*, pages 1028–1033, 2010.

[127] W. Hill, L. Stead, M. Rosenstein, and G. Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 194–201. ACM Press/Addison-Wesley Publishing Co., 1995.

[128] P. Honey, A. Mumford, et al. The manual of learning styles. *Peter Honey Maidenhead*, 1992.

[129] Y. Hong and L. Zhuanyun. A collaborative filtering recommendation algorithm based on forgetting curve. *Journal of Nanjing University*, 46(5):520–527, 2010.

[130] M. F. Hornick, E. Marcadue, and S. Venkayala. *Java data mining - strategy, standard, and practice - A Practical Guide for Architecture, Design and Implementation*. Morgan Kaufmann Publishers, 2007.

[131] S.-L. Huang. Designing utility-based recommender systems for e-commerce: Evaluation of preference-elicitation methods. *Electronic Commerce Research and Applications, Elsevier*, 10(4):398–407, 2011.

[132] Y.-M. Huang, T.-C. Huang, K.-T. Wang, and W.-Y. Hwang. A markov-based recommendation model for exploring the transfer of learning on the web. *Journal of Educational Technology & Society, JSTOR*, 12(2):144–162, 2009.

[133] N. Hurley and M. Zhang. Novelty and diversity in top-n recommendation–analysis and evaluation. *ACM Transactions on Internet Technology (TOIT)*, 10(4):14, 2011.

[134] L. Iaquinta, M. De Gemmis, P. Lops, G. Semeraro, M. Filannino, and P. Molino. Introducing serendipity in a content-based recommender system. In *Eighth International Conference on Hybrid Intelligent Systems (HIS'08)*, pages 168–173. IEEE, 2008.

[135] D. Ifenthaler and C. Schumacher. Learning analytics im hochschulkontext. *WiSt-Wirtschaftswissenschaftliches Studium, Verlag Franz Vahlen GmbH*, 45(4):176–181, 2016.

[136] D. Ifenthaler and C. Schumacher. Student perceptions of privacy principles for learning analytics. *Educational Technology Research and Development, Springer*, 64(5):923–938, 2016.

[137] A. N. Islam and N. Azad. Satisfaction and continuance with a learning management system: Comparing perceptions of educators and students. *The International Journal of Information and Learning Technology, Emerald Group Publishing Limited*, 32(2):109–123, 2015.

[138] K. Jack, J. Hammerton, D. Harvey, J. J. Hoyt, J. Reichelt, and V. Henning. Mendeleys reply to the datatel challenge. *Procedia Computer Science Journal, Elsevier*, 1(2):1–3, 2010.

[139] J. Janssen, C. Tattersall, W. Waterink, B. Van den Berg, R. Van Es, C. Bolman, and R. Koper. Self-organising navigational support in lifelong learning: how predecessors can lead the way. *Computers & Education, Elsevier*, 49(3):781–793, 2007.

[140] T. S. Jastrzembski, K. A. Gluck, and S. Rodgers. The predictive performance optimizer: An adaptive analysis cognitive tool for performance prediction. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 53, pages 1642–1646. SAGE Publications, 2009.

[141] H. Jeon, T. Kim, and J. Choi. Mobile semantic search using personal preference filtering. In *Fourth International Conference on Networked Computing and Advanced Information Management (NCM'08)*, volume 2, pages 531–534. IEEE, 2008.

[142] T. Q. Jiang and W. Lu. Improved slope one algorithm based on time weight. In *Instruments, Measurement, Electronics and Information Engineering*, volume 347 of *Applied Mechanics and Materials*, pages 2365–2368. Trans Tech Publications, 10 2013.

[143] H. Kaatrakoski, A. Littlejohn, and N. Hood. Rethinking professional learning in higher education: a study on how the use of open educational resources triggers the adoption of open educational practice. *Qwerty-Open and Interdisciplinary Journal of Technology, Culture and Education, In Press*, 2017.

[144] F. Kamiran, T. Calders, and M. Pechenizkiy. Techniques for discrimination-free predictive models. In *Discrimination and Privacy in the Information Society*, pages 223–239. Springer, 2013.

[145] M. Kantardzic. *Data mining: concepts, models, methods, and algorithms*. John Wiley & Sons, 2011.

[146] P. B. Kantor, L. Rokach, F. Ricci, and B. Shapira. *Recommender systems handbook*. Springer, 2011.

[147] G. Karypis. Evaluation of item-based top-n recommendation algorithms. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 247–254. ACM, 2001.

[148] H. Kautz, B. Selman, and M. Shah. Referral web: combining social networks and collaborative filtering. *Communications of the ACM*, 40(3):63–65, 1997.

[149] J. M. Kevan and P. R. Ryan. Experience api: Flexible, decentralized and activity-centric data collection. *Technology, Knowledge and Learning, Springer*, 21(1):143–149, 2016.

[150] M. K. Khrib, M. Jemn, and O. Nasraoui. Automatic recommendations for e-learning personalization based on web usage mining techniques and information retrieval. In *Eighth IEEE International Conference on Advanced Learning Technologies (ICALT'08)*, pages 241–245. IEEE, 2008.

[151] E. Kim, S. Pyo, E. Park, and M. Kim. An automatic recommendation scheme of TV program contents for (IP)tv personalization. *Transactions on Broadcasting, IEEE*, VOL. 57, NO. 3:674–684, 2011.

[152] H. Kim, J. Kim, and J. Herlocker. Feature-based prediction of unknown preferences for nearest-neighbor collaborative filtering. *Proceddings of the Fourth IEEE International Conference on Data Mining*, IV, 2004.

[153] E. King and R. Boyatt. Exploring factors that influence adoption of e-learning within higher education. *British Journal of Educational Technology, Wiley Online Library*, 46(6):1272–1280, 2015.

[154] D. L. Kirkpatrick. Techniques for evaluating training programs. *Training and development journal*, 1979.

[155] R. F. Kizilcec, C. Piech, and E. Schneider. Deconstructing disengagement: analyzing learner subpopulations in massive open online courses. In *Proceedings of the third international conference on learning analytics and knowledge*, pages 170–179. ACM, 2013.

[156] M. Kloft, F. Stiehler, Z. Zheng, and N. Pinkwart. Predicting mooc dropout over weeks using machine learning methods. In *Proceedings of the EMNLP 2014 Workshop on Analysis of Large Scale Social Interaction in MOOCs*, pages 60–65, 2014.

[157] S. C. Kong. Developing information literacy and critical thinking skills through domain knowledge learning in digital classrooms: An experience of practicing flipped classroom strategy. *Computers & Education, Elsevier*, 78:160–173, 2014.

[158] Y. Koren. The bellkor solution to the netflix grand prize. *Netflix prize documentation*, 81, 2009.

[159] Y. Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 53(4):89–97, 2010.

[160] G. Koutrika, B. Bercovitz, F. Kaliszan, H. Liou, and H. Garcia-Molina. Courserank: A closed-community social system through the magnifying glass. In *The International Conference on Weblogs and Social Media (ICWSM), AAAI*, 2009.

[161] C. Krauss. Smart learning: Time-dependent context-aware learning object recommendations. *Proceedings of The 29th AAAI International Florida AI Research Society Conference (FLAIRS-29), AAAI*, May 2016.

[162] C. Krauss. A time-dependent recommender system for the prediction of appropriate learning objects. *Doctoral Consortium of the 1st German Learning Analytics Summer Institute*, 1(1), 2017.

[163] C. Krauss and S. Arbanowski. Social preference ontologies for enriching user and item data in recommendation systems. *IEEE International Conference on Data Mining Workshop (ICDMW)*, 2014.

[164] C. Krauss, S. Braun, and S. Arbanowski. Preference ontologies based on social media for compensating the cold start problem. In *SNAKDD'14*, pages 1–8. ACM, 2014.

[165] C. Krauss, R. Chandru, and S. Arbanowski. Towards time-dependent context-sensitive user data for recommending learning objects. *The 11th Reasoning Web Summer School (RW2015) as part of The 9th International Conference on Web Reasoning and Rule Systems (RR2015)*, 2015.

[166] C. Krauss, R. Chandru, A. Merceron, T.-S. An, M. Zwicklbauer, and S. Arbanowski. You might have forgotten this learning content! how the smart learning recommender predicts appropriate learning objects. *International Journal On Advances in Intelligent Systems IntSys16v9n34*, 9(34):472–484, 2016.

[167] C. Krauss, L. George, and S. Arbanowski. TV predictor: personalized program recommendations to be displayed on smarttvs. In *KDD Workshop on Big Data Mining (BigMine'13)*, pages 63–70. ACM, 2013.

[168] C. Krauss, A. Merceron, T.-S. An, M. Zwicklbauer, and S. Arbanowski. The smart learning approach - a mobile learning companion application. *Proceedings of The Eighth International Conference on Mobile, Hybrid, and On-line Learning (eLmL 2016)*, 2016.

[169] C. Krauss, A. Merceron, T.-S. An, M. Zwicklbauer, S. Steglich, and S. Arbanowski. Teaching advanced web technologies with a mobile learning companion application. In *Proceedings of The 16th ACM World Conference on Mobile and Contextual Learning (mLearn 2017), ACM, Larnaca, Cyprus*, 2017.

[170] V. Kumar, J. Nesbit, and K. Han. Rating learning object quality with distributed bayesian belief networks: The why and the how. In *Fifth IEEE International Conference on Advanced Learning Technologies (ICALT)*, pages 685–687. IEEE, 2005.

[171] X. N. Lam, T. Vu, T. D. Le, and A. D. Duong. Addressing cold-start problem in recommendation systems. In *Proceedings of the 2^{nd} international conference on Ubiquitous information management and communication*, pages 208–211. ACM, 2008.

[172] N. Lathia, S. Hailes, and L. Capra. Temporal collaborative filtering with adaptive neighbourhoods. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 796–797. ACM, 2009.

[173] P. Lehsten, R. Zender, U. Lucke, and D. Tavangarian. A service-oriented approach towards context-aware mobile learning management systems. In *2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pages 268–273. IEEE, 2010.

[174] D. Lemire, H. Boley, S. McGrath, and M. Ball. Collaborative filtering and inference rules for context-aware learning object recommendation. *Interactive Technology and Smart Education, Emerald Group Publishing Limited*, 2(3):179–188, 2005.

[175] D. Lemire and A. Maclachlan. Slope one predictors for online rating-based collaborative filtering. *Proceedings of SIAM Data Mining (SDM'05)*, 2005.

[176] C.-T. Li, S.-D. Lin, and M.-K. Shan. Exploiting endorsement information and social influence for item recommendation. In *Proceedings of the 34^{th} international ACM SIGIR conference on Research and development in Information Retrieval*, pages 1131–1132. ACM, 2011.

[177] J. Li, L. Sun, and J. Wang. A slope one collaborative filtering recommendation algorithm using uncertain neighbors optimizing. In *International Conference on Web-Age Information Management*, pages 160–166. Springer, 2011.

[178] L. Li, Y. Zheng, H. Ogata, and Y. Yano. A framework of ubiquitous learning environment. In *The Fourth International Conference on Computer and Information Technology (CIT'04)*, pages 345–350. IEEE, 2004.

[179] R. Likert. A technique for the measurement of attitudes. *Archives of psychology*, 1932.

[180] G. Linden, B. Smith, and J. York. Amazon.com recommendations - item-to-item collaborative filtering. *IEEE Internet Computing*, pages 76–80, 2003.

[181] G. D. Linden, J. A. Jacobi, and E. A. Benson. Collaborative recommendations using item-to-item similarity mappings, 2001. US Patent 6,266,649.

[182] G. Liu, H. Jiang, R. Geng, and H. Li. Application of multidimensional association rules in personal financial services. *IEEE - International Conference On Computer Design And Appliations*, pages 500–503, 2010.

[183] Y. Liu, Z. Xu, B. Shi, and B. Zhang. Time-based k-nearest neighbor collaborative filtering. In *12^{th} International Conference on Computer and Information Technology (CIT), IEEE*, pages 1061–1065. IEEE, 2012.

[184] Z. Liu, H. Wang, W. Qu, and W. Liu. Sparse matrix prediction filling in collaborative filtering. *IEEE International Conference on Scalable Computing and Communications; The Eighth International Conference on Embedded Computing;*, pages 304–307, 2009.

[185] Z. Lu, D. Agarwal, and I. S. Dhillon. A spatio-temporal approach to collaborative filtering. In *Proceedings of the third ACM conference on Recommender systems*, pages 13–20. ACM, 2009.

[186] C. Luo, W. Pang, Z. Wang, and C. Lin. Hete-cf: Social-based collaborative filtering recommendation using heterogeneous relations. In *IEEE International Conference on Data Mining (ICDM)*, pages 917–922. IEEE, 2014.

[187] T. Mahmood and F. Ricci. Improving recommender systems with adaptive conversational strategies. In *Proceedings of the 20th ACM conference on Hypertext and hypermedia*, pages 73–82. ACM, 2009.

[188] A. Malpani, B. Ravindran, and H. Murthy. Personalized intelligent tutoring system using reinforcement learning. In *International FLAIRS Conference of the The Florida Artificial Intelligence Research Society (FLAIRS'2011), AAAI*, 2011.

[189] A. Maniezzo. Distributed optimization by ant colonies. In *Toward a practice of autonomous systems: proceedings of the First European Conference on Artificial Life*, page 134. Mit Press, 1992.

[190] N. Manouselis, H. Drachsler, K. Verbert, and E. Duval. *Recommender systems for learning*. Springer Science & Business Media, 2012.

[191] N. Manouselis, H. Drachsler, R. Vuorikari, H. Hummel, and R. Koper. Recommender systems in technology enhanced learning. In *Recommender systems handbook*, pages 387–415. Springer, 2011.

[192] N. Manouselis, T. Kosmopoulos, and K. Kastrantas. Developing a recommendation web service for a federation of learning repositories. In *International Conference on Intelligent Networking and Collaborative Systems (INCOS'09), IEEE*, pages 208–211. IEEE, 2009.

[193] N. Manouselis, R. Vuorikari, and F. Van Assche. Simulated analysis of maut collaborative filtering for learning object recommendation. In *Proceedings of the 1st Workshop on Social Information Retrieval for Technology Enhanced Learning*, pages 27–35, 2007.

[194] P. Massa and P. Avesani. Trust-aware recommender systems. In *Proceedings of the 2007 ACM conference on Recommender systems*, pages 17–24. ACM, 2007.

[195] L. P. Michel. Digitale bildung auf dem weg ins jahr 2025. *mmb Institut – Gesellschaft fuer Medien- und Kompetenzforschung mbH*, 2016.

[196] S. E. Middleton, N. R. Shadbolt, and D. C. De Roure. Ontological user profiling in recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):54–88, 2004.

[197] B. N. Miller, I. Albert, S. K. Lam, J. A. Konstan, and J. Riedl. Movielens unplugged: experiences with an occasionally connected recommender system. In *Proceedings of the 8^{th} international conference on Intelligent user interfaces*, pages 263–266. ACM, 2003.

[198] M. A. Morid, M. Shajari, and A. H. Golpayegani. Who are the most influential users in a recommender system? In *Proceedings of the 13^{th} international conference on electronic commerce*, page 19. ACM, 2011.

[199] A. H. Nabizadeh, A. Mário Jorge, and J. Paulo Leal. Rutico: Recommending successful learning paths under time constraints. In *Adjunct Publication of the 25th Conference on User Modeling, Adaptation and Personalization*, pages 153–158. ACM, 2017.

[200] R. J. Nadolski, B. Van den Berg, A. J. Berlanga, H. Drachsler, H. G. Hummel, R. Koper, and P. B. Sloep. Simulating light-weight personalised recommender systems in learning

networks: A case for pedagogy-oriented and rating-based hybrid recommendation strategies. *Journal of Artificial Societies and Social Simulation*, 12(1):4, 2009.

[201] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology, Elsevier*, 48(3):443–453, 1970.

[202] F. Neven and E. Duval. Reusable learning objects: a survey of lom-based repositories. In *Proceedings of the tenth ACM international conference on Multimedia*, pages 291–294. ACM, 2002.

[203] K. Niemann, M. Scheffel, and M. Wolpers. An overview of usage data formats for recommendations in tel. In *2 nd Workshop on Recommender Systems for Technology Enhanced Learning (RecSysTEL 2012)*, page 95, 2012.

[204] D. F. Onah, J. Sinclair, and R. Boyatt. Dropout rates of massive open online courses: behavioural patterns. *Proceedings of the EDULEARN 2014, IATED Academy*, pages 5825–5834, 2014.

[205] R. Oppermann. Adaptively supported adaptability. *International Journal of Human-Computer Studies, Elsevier*, 40(3):455–472, 1994.

[206] H. T. Ozturk, D. Deryakulu, H. Ozcinar, and D. Atal. Advancing learning analytics in online learning environments through the method of sequential analysis. In *2014 International Conference on Multimedia Computing and Systems (ICMCS)*, pages 512–516. IEEE, 2014.

[207] G. Paquette, D. Rogozan, and O. Marino. Competency comparison relations for recommendation in technology enhanced learning scenarios. Proceedings of the RecSysTEL 2012 - CEUR Workshop Proceedings, 2012.

[208] Z. A. Pardos, R. S. Baker, M. O. San Pedro, S. M. Gowda, and S. M. Gowda. Affective states and state tests: Investigating how affect throughout the school year predicts end of year learning outcomes. In *Proceedings of the Third International Conference on Learning Analytics and Knowledge*, pages 117–124. ACM, 2013.

[209] D. H. Park, H. K. Kim, I. Y. Choi, and J. K. Kim. A literature review and classification of recommender systems research. *Expert Systems with Applications, Elsevier*, 39(11):10059–10072, 2012.

[210] S.-T. Park and W. Chu. Pairwise preference regression for cold-start recommendation. In *Proceedings of the third ACM conference on Recommender systems*, pages 21–28. ACM, 2009.

[211] P. I. Pavlik Jr. and J. R. Anderson. Practice and forgetting effects on vocabulary memory: An activation-based model of the spacing effect. *Cognitive Science, Wiley Online Library*, 29(4):559–586, 2005.

[212] M. J. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review, Springer*, 13(5-6):393–408, 1999.

[213] R. Pelanek and P. Jarusek. Student modeling based on problem solving times. *International Journal of Artificial Intelligence in Education, Springer New York*, 25(4):493–519, 2015.

[214] D. Pelleg, A. W. Moore, et al. X-means: Extending k-means with efficient estimation of the number of clusters. In *International Conference on Machine Learning (ICML)*, volume 1, pages 727–734, 2000.

[215] J. Pérez-Marcos and V. L. Batista. Recommender system based on collaborative filtering for spotify's users. In *International Conference on Practical Applications of Agents and Multi-Agent Systems*, pages 214–220. Springer, 2017.

[216] V. Pieterse and P. E. Blacks. Levenshtein distance. *Dictionary of Algorithms and Data Structures*, 2013.

[217] M. Piotte and M. Chabbert. The pragmatic theory solution to the netflix grand prize. *Netflix prize documentation*, 2009.

[218] P. R. Polsani. Use and abuse of reusable learning objects. *Journal of Digital information*, 3(4), 2003.

[219] L. Qian et al. A study on the influence of recommendation models on customer satisfaction in b2C e-commerce. In *2^{nd} International Conference on Networking and Digital Society (ICNDS)*, volume 2, pages 452–455. IEEE, 2010.

[220] R. Rada. Efficiency and effectiveness in computer-supported peer-peer learning. *Computers and Education, Elsevier*, 30(3):137 – 146, 1998.

[221] S. Rafaeli, M. Barak, Y. Dan-Gur, and E. Toch. Qsia – a web-based environment for learning, assessing and knowledge sharing in communities. *Computers & Education, Elsevier*, 43(3):273–289, 2004.

[222] A. M. Rashid, I. Albert, D. Cosley, S. K. Lam, S. M. McNee, J. A. Konstan, and J. Riedl. Getting to know you: learning new user preferences in recommender systems. In *Proceedings of the 7^{th} international conference on Intelligent user interfaces*, pages 127–134. ACM, 2002.

[223] M. M. Recker, A. Walker, and D. Wiley. An interface for collaborative filtering of educational resources. *In Proceedings of the 2000 International Conference on Artificial Intelligence (IC-AI'2000)*, 2000.

[224] M. M. Recker and D. A. Wiley. A non-authoritative educational metadata ontology for filtering and recommending learning objects. *Interactive learning environments, Taylor & Francis*, 9(3):255–271, 2001.

[225] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM, 1994.

[226] P. Resnick and H. R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[227] N. Ryan, J. Pascoe, and D. Morse. Enhanced reality fieldwork: the context aware archaeological assistant. *Bar International Series, Tempus Reparatsm*, 750:269–274, 1999.

[228] A. Said and A. Bellogín. Comparative recommender system evaluation: benchmarking recommendation frameworks. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 129–136. ACM, 2014.

[229] J. R. Sanders. *The program evaluation standards: how to assess evaluations of educational programs*. Journal of Educational Measurement, Sage, 1994.

[230] J. Sang and C. Xu. Social influence analysis and application on multimedia sharing websites. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 9(1s):53, 2013.

[231] E. Sangineto, N. Capuano, M. Gaeta, and A. Micarelli. Adaptive course generation through learning styles representation. *Universal Access in the Information Society, Springer*, 7(1-2):1–23, 2008.

[232] O. C. Santos and J. G. Boticario. Recommendation strategies for promoting elearning performance factors for all. *Proceedings of the 6th Workshop on Intelligent Techniques for Web Personalization & Recommender Systems*, pages 89–98, 2008.

[233] B. S. Sarma and B. Ravindran. Intelligent tutoring systems using reinforcement learning to teach autistic students. In *Home Informatics and Telematics: ICT for The Next Billion*, pages 65–78. Springer, 2007.

[234] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2nd ACM conference on Electronic commerce*, pages 158–167. ACM, 2000.

[235] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *GroupLens Research Group/Army HPC Research Center*, 2001.

[236] R. Schaller, M. Harvey, and D. Elsweiler. Recsys for distributed events: Investigating the influence of recommendations on visitor plans. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 953–956. ACM, 2013.

[237] C. Schatten and L. Schmidt-Thieme. Adaptive content sequencing without domain information. In *International Conference on Computer Supported Education (CSEDU14)*, pages 25–33, 2014.

[238] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260. ACM, 2002.

[239] B. N. Schilit and M. M. Theimer. Disseminating active map information to mobile hosts. *IEEE Network, IEEE*, 8(5):22–32, 1994.

[240] A. Schmidt. Impact of context-awareness on the architecture of learning support systems. In *Architecture solutions for E-Learning systems*, pages 306–319. IGI Global, 2008.

[241] B. Schwartz. The paradox of choice. Harper Collins New York, NY, 2004.

[242] G. Shani and A. Gunawardana. Evaluating recommendation systems. In *Recommender systems handbook*, pages 257–297. Springer, 2011.

[243] U. Shardanand and P. Maes. Social information filtering: algorithms for automating ?word of mouth? In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217. ACM Press/Addison-Wesley Publishing Co., 1995.

[244] L.-p. Shen and R.-m. Shen. Learning content recommendation service based-on simple sequencing specification. In *Advances in Web-Based Learning–ICWL 2004*, pages 363–370. Springer, 2004.

[245] C. Shin and W. Woo. Socially aware TV program recommender for multiple viewers. *Transactions on Consumer Electronics, IEEE*, pages 927–932, 2009.

[246] T. J. Shuell. Phases of meaningful learning. *Review of educational research, Sage Publications Sage CA: Thousand Oaks, CA*, 60(4):531–547, 1990.

[247] M.-Á. Sicilia, E. García-Barriocanal, S. Sánchez-Alonso, and C. Cechinel. Exploring user-based recommender results in large learning object repositories: the case of merlot. *Procedia Computer Science, Elsevier*, 1(2):2859–2864, 2010.

[248] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of molecular biology, Elsevier*, 147(1):195–197, 1981.

[249] J. Stamper, A. Niculescu-Mizil, S. Ritter, G. Gordon, and K. Koedinger. Algebra i 2008-2009. *Challenge data set from KDD Cup 2010 Educational Data Mining Challenge. Find it at http://pslcdatashop.web.cmu.edu/KDDCup/downloads.jsp*, 2010.

[250] H. Steck, R. van Zwol, and C. Johnson. Interactive recommender systems: Tutorial. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 359–360. ACM, 2015.

[251] H. Stern, R. Kaiser, P. Hofmair, P. Kraker, S. N. Lindstaedt, and P. Scheir. Content recommendation in aposdle using the associative network. *Journal of Universal Computer Scence (J.UCS)*, 16(16):2214–2231, 2010.

[252] S. S. Stevens. *On the Theory of Scales of Measurement.* Science, American Association for the Advancement of Science, AAAS, 1946.

[253] X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence, Hindawi Publishing Corp.*, 2009:4, 2009.

[254] W. e. a. Swartout. Designing a personal assistant for life-long learning. *Proceedings of 29th International FLAIRS Conference of the The Florida Artificial Intelligence Research Society, AAAI, Key Largo, Florida, USA*, 2016.

[255] K. Swearingen and R. Sinha. Beyond algorithms: An hci perspective on recommender systems. In *SIGIR 2001 Workshop on Recommender Systems*, volume 13, pages 1–11. ACM, 2001.

[256] C. Swertz, A. Schmoelz, A. Forstner, F. Heberle, P. Henning, A. Streicher, B. A. Bargel, J. Bock, and S. Zander. A pedagogical ontology as a playground in adaptive elearning environments. *Proceedings of the Informatik 2013 conference, Lecture Notes in Informatics P-220 (2014) p. 1955-1960*, 2014.

[257] M. Szabo and K. Flesher. Cmi theory and practice: Historical roots of learning management systems. *ERIC*, 2002.

[258] S. Taneja and A. Goel. Mooc providers and their strategies. *International Journal of Computer Science and Mobile Computing*, 3(5):222–228, 2014.

[259] T. Tang and G. McCalla. Smart recommendation for an evolving e-learning system: Architecture and experiment. *International Journal on E-learning*, 4(1):105–129, 2005.

[260] T. Y. Tang, P. Winoto, and K. C. Chan. On the temporal analysis for improved hybrid recommendations. In *International Conference on Web Intelligence (WI 2003), IEEE/WIC*, pages 214–220. IEEE, 2003.

[261] T. T. Tanimoto. Elementary mathematical theory of classification and prediction. *International Business Machines Corp.*, 1958.

[262] N. Thai-Nghe, L. Drumond, A. Krohn-Grimberghe, and L. Schmidt-Thieme. Recommender system for predicting student performance. *Procedia Computer Science, Elsevier*, 1(2):2811–2819, 2010.

[263] W. Thalheimer. How much do people forget? *Work-Learning Research, Inc. Somerville, MA, USA*, 2010.

[264] N. Tintarev and J. Masthoff. Designing and evaluating explanations for recommender systems. In *Recommender Systems Handbook*, pages 479–510. Springer, 2011.

[265] A. Töpfer. *Erfolgreich Forschen: Ein Leitfaden für Bachelor-, Master-Studierende und Doktoranden*. Springer-Verlag, 2012.

[266] A. Töscher, M. Jahrer, and R. M. Bell. The bigchaos solution to the netflix grand prize. *Netflix prize documentation*, 2009.

[267] K. H. Tsai, T. K. Chiu, M. C. Lee, T. Wang, et al. A learning objects recommendation model based on the preference and ontological approaches. In *Advanced Learning Technologies, 2006. Sixth International Conference on*, pages 36–40. IEEE, 2006.

[268] M. Van Setten, S. Pokraev, and J. Koolwaaij. Context-aware recommendations in the mobile tourist application compass. In *Adaptive hypermedia and adaptive web-based systems*, pages 515–548. Springer, 2004.

[269] K. Verbert, H. Drachsler, N. Manouselis, M. Wolpers, R. Vuorikari, and E. Duval. Dataset-driven research for improving recommender systems for learning. In *Proceedings of the 1st International Conference on Learning Analytics and Knowledge*, pages 44–53. ACM, 2011.

[270] K. Verbert, N. Manouselis, X. Ochoa, M. Wolpers, H. Drachsler, I. Bosnic, and E. Duval. Context-aware recommender systems for learning: a survey and future challenges. *IEEE Transactions on Learning Technologies*, 5(4):318–335, 2012.

[271] A. N. Viet and D. H. Si. Acgs: Adaptive course generation system-an efficient approach to build e-learning course. In *The Sixth IEEE International Conference on Computer and Information Technology (CIT'06)*, pages 259–259. IEEE, 2006.

[272] M. Vlachos, C. Meek, Z. Vagena, and D. Gunopulos. Identifying similarities, periodicities and bursts for online search queries. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 131–142. ACM, 2004.

[273] L. Voß, C. Schatten, C. Mazziotti, and L. Schmidt-Thieme. A transfer learning approach for applying matrix factorization to small its datasets. *International Educational Data Mining Society, ERIC*, 2015.

[274] E. Vozalis and K. G. Margaritis. Analysis of recommender systems algorithms. In *The 6th Hellenic European Conference on Computer Mathematics & its Applications*, pages 732–745, 2003.

[275] R. Vuorikari and R. Koper. Ecology of social search for learning resources. *Campus-Wide Information Systems, Emerald Group Publishing Limited*, 26(4):272–286, 2009.

[276] R. Vuorikari and X. Ochoa. Exploratory analysis of the main characteristics of tags and tagging of educational resources in a multi-lingual context. *Journal of Digital Information*, 10(2), 2009.

[277] L. S. Vygotsky. *Mind in society: The development of higher psychological processes*. Harvard university press, 1980.

[278] P. Wang and H. W. Ye. A personalized recommendation algorithm combining slope one scheme and user based collaborative filtering. *International Conference on Industrial and Information Systems*, pages 152–154, 2009.

[279] R. Wang, F. Chen, Z. Chen, T. Li, G. Harari, S. Tignor, X. Zhou, D. Ben-Zeev, and A. T. Campbell. Studentlife: assessing mental health, academic performance and behavioral trends of college students using smartphones. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 3–14. ACM, 2014.

[280] W. R. Watson and S. L. Watson. What are learning management systems, what are they not, and what should they become. *TechTrends, Springer*, 51(2):29, 2007.

[281] S. Weibelzahl. Evaluation of adaptive systems. *User Modeling, Springer*, pages 292–294, 2001.

[282] F. Wolinski, F. Vichot, and M. Stricker. Using learning-based filters to detect rule-based filtering obsolescence. In *Content-Based Multimedia Information Access-Volume 2*, pages 1208–1220. Le Centre de Hautes Etudes Internationales d'informatique documentaire, 2000.

[283] M. Wolpers and K. Niemann. datatel challenge: Cam for mace. In *1st Workshop on Recommender Systems for Technology Enhanced Learning (RecSysTEL 2010)*, 2010.

[284] B. P. Woolf, V. Shute, K. VanLehn, W. Burleson, J. L. King, D. Suthers, B. Bredeweg, R. Luckin, R. S. J. D. Baker, and E. Tonkin. A roadmap for education technology. *Computing Community Consortium, NSF, Computing Research Association*, 2010.

[285] B. R. Worthen, J. R. Sanders, and J. L. Fitzpatrick. Program evaluation. *Alternative approaches and practical guidelines, Prentice Hall*, 2, 1997.

[286] L. Xiang and Q. Yang. Time-dependent models in collaborative filtering based recommender system. In *IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technologies (WI-IAT'09), IEEE*, volume 1, pages 450–457. IET, 2009.

[287] L. Xiang, Q. Yuan, S. Zhao, L. Chen, X. Zhang, Q. Yang, and J. Sun. Temporal recommendation on graphs via long-and short-term preference fusion. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 723–732. ACM, 2010.

[288] X. Xiong, Y. Wang, and J. B. Beck. Improving students' long-term retention performance: a study on personalized retention schedules. In *Proceedings of the Fifth International Conference on Learning Analytics And Knowledge*, pages 325–329. ACM, 2015.

[289] J. A. Xu and K. Araki. A SVM-based personal recommendation system for TV programs. *12th International Multi-Media Modelling Conference Proceedings, IEEE*, pages 401–404, 2006.

[290] D. Yang, D. Zhang, Z. Yu, and Z. Wang. A sentiment-enhanced personalized location recommendation system. In *Proceedings of the 24th ACM Conference on Hypertext and Social Media (HT13), ACM*, pages 119–128, 2013.

[291] Y. J. Yang and C. Wu. An attribute-based ant colony system for adaptive learning object recommendation. *Expert Systems with Applications, Elsevier*, 36(2):3034–3047, 2009.

[292] W. Yanping and C. Yan. Psychology reactance to online recommendations: The influence of time pressure. In *3rd International Conference on System Science, Engineering Design and Manufacturing Informatization (ICSEM)*, volume 1, pages 347–350. IEEE, 2012.

[293] J. Yau and M. Joy. A context-aware and adaptive learning schedule framework for supporting learners' daily routines. In *Second International Conference on Systems (ICONS'07)*, pages 31–31. IEEE, 2007.

[294] M. Ye, X. Liu, and W.-C. Lee. Exploring social influence for recommendation: a generative model approach. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 671–680. ACM, 2012.

[295] X. Yi, L. Hong, E. Zhong, N. N. Liu, and S. Rajan. Beyond clicks: dwell time for personalization. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 113–120. ACM, 2014.

[296] A. M. F. Yousef, M. A. Chatti, U. Schroeder, M. Wosnitza, and H. Jakobs. Moocs-a review of the state-of-the-art. In *Proceedings of the 6th International Conference on Computer Supported Education (CSEDU)*, pages 9–20, 2014.

[297] H. Yu and Z. Li. A collaborative filtering method based on the forgetting curve. In *2010 International Conference on Web Information Systems and Mining (WISM)*, volume 1, pages 183–187. IEEE, 2010.

[298] K. Yu, A. Schwaighofer, V. Tresp, X. Xu, and H.-P. Kriegel. Probabilistic memory-based collaborative filtering. *Transactions on Knowledge and Data Engineering, IEEE*, VOL. 16, NO. 1:56–69, 2004.

[299] H. Zhang and S. Zheng. Personalized TV program recommendation based on TV-anytime metadata. *IEEE Consumer Electronics. Proceedings of the Ninth International Symposium*, pages 242–246, 2005.

[300] Y. Zhang and Y. Liu. A collaborative filtering algorithm based on time period partition. In *Third International Symposium on Intelligent Information Technology and Security Informatics (IITSI)*, pages 777–780. IEEE, 2010.

[301] Q. Zhao, S. C. Hoi, T.-Y. Liu, S. S. Bhowmick, M. R. Lyu, and W.-Y. Ma. Time-dependent semantic similarity measure of queries using historical click-through data. In *Proceedings of the 15th international conference on World Wide Web*, pages 543–552. ACM, 2006.

[302] J. Zhou and T. Luo. Towards an introduction to collaborative filtering. *International Conference on Computational Science and Engineering (CSE'09), IEEE*, pages 576–581, 2009.

[303] J. Zhou and T. Luo. A novel approach to solve the sparsity problem in collaborative filtering. *International Conference on Networking, Sensing and Control (ICNSC), IEEE*, pages 165–170, 2010.

[304] K. G. Zografos and K. N. Androutsopoulos. Algorithms for itinerary planning in multimodal transportation networks. *IEEE Transactions on Intelligent Transportation Systems*, 9(1):175–184, 2008.

[305] M. Zwickelbauer, C. Krauss, A. Merceron, J. P. Kania, and M. Scharp. Smart learning: Der digitale lernbegleiter für die berufliche bildung. *Proceedings of DeLFI 2015, Gesellschaft für Informatik Publisher, Munich*, 2015.

A. Publications

The following works were published during the development of this dissertation. All publications were significantly influenced by the author.

A.1. International Peer-Reviewed Papers

- Krauss, Christopher; Merceron, Agathe; An, Truong-Sinh; Zwicklbauer, Miggi; Steglich, Stephan; Arbanowski, Stefan. *Teaching Advanced Web Technologies with a Mobile Learning Companion Application.* In: Proceedings of The 16th ACM World Conference on Mobile and Contextual Learning (mLearn 2017), ACM, October 30 - November 1, 2017, Larnaca, Cyprus. `https://doi.org/10.1145/3136907.3136937`
- An, Truong-Sinh; Krauss, Christopher; Merceron, Agathe. *Can Typical Behaviors Identified in MOOCs be Discovered in Other Courses?.* In: Proceedings of The 10th International Conference on Educational Data Mining (EDM 2017), 25 - 28 June 2017, Wuhan, China. `http://educationaldatamining.org/EDM2017/proc_files/papers/paper_58.pdf`
- Pham, Stefan; Krauss, Christopher; Silhavy, Daniel; Arbanowski, Stefan. *Personalized Dynamic Ad Insertion with MPEG DASH.* In: Proceedings of the IEEE Asia Pacific Conference on Multimedia and Broadcasting (APMediaCast 2016), 17-19 November 2016, Bali, Indonesia. `https://doi.org/10.1109/APMediaCast.2016.7878162`
- Bayat, Berken; Krauss, Christopher; Merceron, Agathe; Arbanowski, Stefan. *Supervised Speech Act Classification of Messages in German Online Discussions.* In: Proceedings of The 29th AAAI International Florida AI Research Society Conference (FLAIRS-29), AAAI, May 16 - 18 2016, Key Largo, USA. `http://www.aaai.org/ocs/index.php/FLAIRS/FLAIRS16/paper/view/12950`
- Krauss, Christopher. *Smart Learning: Time-dependent Context-aware Learning Object Recommendations.* Short Paper and Poster in: Proceedings of The 29th AAAI International Florida AI Research Society Conference (FLAIRS-29), AAAI, May 16 - 18 2016, Key Largo, USA. `http://www.aaai.org/ocs/index.php/FLAIRS/FLAIRS16/paper/view/12955` (Best Poster Nominee)
- Krauss, Christopher; Merceron, Agathe; An, Truong-Sinh; Zwicklbauer, Miggi; Arbanowski, Stefan. *The Smart Learning Approach - A mobile Learning Companion Application.* In:

Proceedings of The Eighth International Conference on Mobile, Hybrid, and On-line Learning (eLmL 2016). IARIA, April 24 - 28, 2016 - Venice, Italy. http://www.thinkmind.org/index.php?view=article&articleid=elml_2016_1_30_50091 (Best Paper Award)

- Zwicklbauer, Miggi; Merceron, Agathe; Krauss, Christopher; Dinziol, Martin; Scharp, Michael. *Smart Learning: Der digitale Lernbegleiter für die berufliche Bildung.* In: Proceedings of 13. e-Learning Fachtagung Informatik der Fachgruppe E-Learning der Gesellschaft für Informatik e.V. (DeLFI 2015): Forschungsbeitrag, 1 – 4 September 2015, Munic, Germany. http://subs.emis.de/LNI/Proceedings/Proceedings247/article15.html
- Seeliger, Robert; Krauss, Christopher; Wilson, Annette; Zwicklbauer, Miggi; Arbanowski, Stefan. *Towards Personalized Smart City Guide Services in Future Internet Environments.* In: Proceedings of the 24th International Conference on World Wide Web Companion, IEEE. Florence, Italy, May 18th 2015. https://doi.org/10.1109/INNOVATIONS.2015.7381548
- Krauss, Christopher; Arbanowski, Stefan. *Social Preference Ontologies for Enriching User and Item Data in Recommendation Systems.* In: Proceedings of The 2014 IEEE International Conference on Data Mining Workshop (ICDMW 2014), IEEE, 14 December 2014, Shenzhen, China. http://sentic.net/sentire/2014/krauss.pdf
- Krauss, Christopher; Braun, Sascha; Arbanowski, Stefan. *Preference Ontologies based on Social Media for compensating the Cold Start Problem.* In: Proceedings of The 8th International Workshop on Social Network Mining and Analysis (SNAKDD). ACM, 23th August 2014, New York. http://dl.acm.org/citation.cfm?doid=2659480.2659504
- Krauss, Christopher; Seeliger, Robert; Wilson, Annette; Arbanowski, Stefan. *Enriched personalized multi-screen content for social connected TV.* In: Proceedings of the International Conference On Interactive Experiences for Television and online Video (TVX2014). ACM, 25th-27th June 2014, Newcastle. Published via figshare: https://figshare.com/articles/WP_108_Enriched_personalized_multi_screen_content_for_social_connected_TV/1032590
- Krauss, Christopher; Bassbouss, Louay; Pham, Stefan; Kaiser, Stefan; Arbanowski, Stefan; Steglich, Stephan. *Challenges for enabling targeted multi-screen advertisement for interactive TV services.* In: Proceedings of The Fourth W3C Web and TV Workshop. W3C, 12–13 March 2014, Munich, Germany. http://www.w3.org/2013/10/tv-workshop/papers.html
- Krauss, Christopher; George, Lars; Arbanowski, Stefan. 2013. *TV Predictor: personalized program recommendations to be displayed on SmartTVs.* In: Proceedings of the 2nd International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications (BigMine '13). ACM, New York, NY, USA, 63-70. DOI=10.1145/2501221.2501230 http://doi.acm.org/10.1145/2501221.2501230
- George, Lars; Kaiser, Stefan; Pham, Stefan; Krauss, Christopher. *Internet-Delivered Television using MPEG DASH: Opportunities and Challenges.* In: Proceedings of the 67th

Broadcast Engineering Conference (BEC) at National Association of Broadcasters (NAB). Las Vegas, NV, USA, 67 / 2013 / 20131201748. Printed Version only.

A.2. Patent Pending

- Krauss, Christopher; Seeliger, Robert; Arbanowski, Stefan; Andrea Duerager; Reisser, Lukas; Hinterstoisser, Sebastian. *Method, Apparatus, Computer Program and System for Determining Information Related to the Audience of an Audio-Visual Content Program.* Request for grant of a European patent, Application No FHG170602PAEP; Applicant 1: Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e.V.; Applicant 2: Red Bull GmbH.

A.3. Other Publications and Contributions

- Krauss, Christopher; Schlösser, Holger. *Smart Learning — neue Ansätze für individuelles Lernen.* Invited Article for the SQ Magazin – Fachzeitschrift für Software-Qualität, Schwerpunkt: Die Zukunft der Weiterbildung, Vol. 45, December 2017.
- An, Truong-Sinh; Merceron, Agathe; Krauss, Christopher. *Können typische Lernverhalten, wie sie in MOOCs gefunden wurden, auch in anderen Kursen auftauchen?* In: Proceedings of the Research Day, Beuth Hochschule für Technik Berlin, 2017, Berlin, Germany
- Krauss, Christopher. *A Time-Dependent Recommender System for the Prediction of Appropriate Learning Objects.* Doctoral Consortium of the 1st German Learning Analytics Summer Institute 2017, HTW Berlin, April 6 – April 7 2017, Berlin, Germany.
- Krauss, Christopher; Merceron, Agathe; An, Truong-Sinh; Zwicklbauer, Miggi; Dinziol, Martin; Scharp, Michael. *„Smart Learning“: ein Lernempfehlungssystem für digitale Medien.* In: Learntec 2017, 24 - 26 January 2017, Karlsruhe, Germany.
- Krauss, Christopher. *Smart Learning – Neuer Ansatz für individuelles Lernen.* Interview in: eLearning Journal 1/2017, Siepmann Media, Januar 2017, Hagen, Germany. Online available: `http://www.elearning-journal.de/index.php?id=1996`
- Krauss, Christopher; Chandru, Rakesh; Merceron, Agathe; An, Truong-Sinh; Zwicklbauer, Miggi; Arbanowski, Stefan. *You Might Have Forgotten This Learning Content! How the Smart Learning Recommender Predicts Appropriate Learning Objects.* In: IntSys16v9n34, International Journal On Advances in Intelligent Systems, 2016 no 3& 4. Xpert Publishing Services, December 31, 2016 - Online
- Krauss, Christopher; Merceron, Agathe; An, Truong-Sinh; Zwicklbauer, Miggi; Dinziol, Martin; Scharp, Michael. *„Smart Learning“: ein digitaler Lernbegleiter für die berufliche Bildung.* In: Proceedings of the Research Day, Beuth Hochschule für Technik Berlin, 2016,

Berlin, Germany

- Zwicklbauer, Miggi; Krauss, Christopher; An, Truong-Sinh; Merceron, Agathe. *Mobiles Lernen in der beruflichen Bildung mit dem digitalen Lernbegleiter.* In: Learntec 2016, 26 - 28 January 2016, Karlsruhe, Germany.
- Krauss, Christopher; Chandru, Rakesh; Arbanowski, Stefan. *Towards Time-dependent Context-sensitive User Data for recommending Learning Objects.* Poster at: The 11th Reasoning Web Summer School (RW2015) as part of The 9th International Conference on Web Reasoning and Rule Systems (RR2015), August 4, 2015, Berlin, Germany.
- Merceron, Agathe; Kania, Jost; Krauss, Christopher; Scharp, Michael; Zwicklbauer, Miggi. *„Smart Learning" für die handwerkliche Weiterbildung.* In: Proceedings of the Research Day, Beuth Hochschule für Technik Berlin, 21 April, 2015, Berlin, Germany.

A.4. Supervised Theses Topics

- Salzmann, Andreas; Manfred (1. Examiner); Magedanz, Thomas (2. Examiner); Krauss, Christopher (Supervisor). *Leveraging Time-Dependent Multi-Modal Routing Plan Algorithms to Create Personalized Learning Paths for Technology Enhanced Learning.* Master Thesis at TU Berlin, August 2, 2017
- Chandru, Rakesh; Samberg, Ulrich (1. Examiner); Lüssem, Jens (2. Examiner); Krauss, Christopher (Supervisor). *Oblivion in Recommender Systems – The forgetting effect in predictions.* Master Thesis at Kiel University of Applied Sciences, 2016
- Häuser, Sarah; Krauss, Christopher (1. Examiner); Schmiedecke, Ilse (2 . Examiner). *Gamification for leveraging Recommendation Engines: Competition and Play Instinct against Data Sparsity.* Master Thesis at Beuth University Berlin, 2015
- Braun, Sascha; Hauswirth, Manfred (1. Examiner); Magedanz, Thomas (2. Examiner); Krauss, Christopher (Supervisor). *Event based Recommendation Systems in Smart Environments.* Master Thesis at TU Berlin, June 18, 2015.
- Bayat, Berken; Hauswirth, Manfred (1. Examiner); Magedanz, Thomas (2. Examiner); Krauss, Christopher (Supervisor). *Supervised Speech Act Classification of Messages in German Online Discussions.* Bachelor Thesis at TU Berlin, May 18, 2015.
- Liwotto, Marius; Hauswirth, Manfred (1. Examiner); Magedanz, Thomas (2. Examiner); Krauss, Christopher (Supervisor). *Trend Detection in Social Networks through Topic and Sentiment Analysis.* Bachelor Thesis at TU Berlin, April 7, 2015.
- An, Truong-Sinh; Popescu-Zeletin, Radu (Examiner); Krauss, Christopher (Supervisor). *Development of a Recommendation as a Service Platform using NoSQL Database Technology.* Diploma at TU Berlin, March 31, 2014.
- Kops, Lars; Wijnhoven, A. (1. Examiner); Zurek, Peter (2 . Examiner); Nölling, Karsten

(Supervisor); Krauss, Christopher (Supervisor). *Improving travel recommenders by benefit segmentation: The case of George.* Dual Master Thesis at TU Berlin and University Twente, November 5, 2013.

A.5. Supervised Seminar Papers and Student Projects

- Cory, Thomas; Diechmann, Henrik; Müller, Julian; Siemund, Alia; Weiss, Laura; Wille, Fiona; Ziehn, Ariane; Krauss, Christopher (Supervisor). *Gamification for Learning Environments - Badges, Communities, Dashboards, Group Quizzes and Ranks.* Project Advanced Web Technologies (Open Distributed Systems) at TU Berlin, WS 2017/18.
- Rezaie, Parisa; Krauss, Christopher (Supervisor). *Pattern Recognition in the Data of the Learning Companion Application.* Project Advanced Web Technologies (Open Distributed Systems) at TU Berlin, WS 2017/18.
- Mikaelyan, Armine; Krauss, Christopher (Supervisor). *Multi-Language Support for the Learning Companion Application.* Project Advanced Web Technologies (Open Distributed Systems) at TU Berlin, WS 2017/18.
- Szücs, Dora; Krauss, Christopher (Supervisor). *Online Survey for Analyzing the Acceptance of the Smart Learning Recommender.* Project Advanced Web Technologies (Open Distributed Systems) at TU Berlin, WS 2017/18.
- Mudgil, Vinay; Shah Jamal, Mohammed; Krauss, Christopher (Supervisor). *Thinking Aloud Studies of the Learning Companion Application.* Project Advanced Web Technologies (Open Distributed Systems) at TU Berlin, WS 2017/18.
- Ciobanu, Alexandru; Krauss, Christopher (Supervisor). *K-Nearest-Neighbour-Algorithm for Prediction of Course Grades.* Project Advanced Web Technologies (Open Distributed Systems) at TU Berlin, SS 2017.
- Awanish, Kumar; Vashisth, Gaurav; Gulaganjihalli, Rudresha; Krauss, Christopher (Supervisor). *Realization and Evaluation of a Time-Aware Recommender System for Lecture Materials.* Project Advanced Web Technologies (Open Distributed Systems) at TU Berlin, SS 2017.
- Conzelmann, Miro; Krauss, Christopher (Supervisor). *Realization and Evaluation of the Time-Weighted Slope One Algorithm for the Recommendation of Learning Objects.* Project Advanced Web Technologies (Open Distributed Systems) at TU Berlin, SS 2017.
- Seggemu, Akarsh; Ochunyi, Philip B.; Huynh, Khac Hieu; Krauss, Christopher (Supervisor). *Evaluation of the Acceptance of a Learning Platform for University Courses.* Project Advanced Web Technologies (Open Distributed Systems) at TU Berlin, WS 2016/17.
- Skowronek, David; Krauss, Christopher (Supervisor). *Slide2LOM - A Powerpoint Exporter for the generation of IMS LOM-compliant learning objects.* Project Advanced Web

Technologies (Open Distributed Systems) at TU Berlin, WS 2016/17.

- Haoua, Marsa; Krauss, Christopher (Supervisor). *Knowledge Discovery in Discussion Threads: Mining of Topical Correlations and Semantic Associations in Social eLearning Forums*. Project Advanced Web Technologies (Open Distributed Systems) at TU Berlin, SS 2016.
- Ott, Martin; Herzog, David; Krauss, Christopher (Supervisor). *Linked Open Data for Learning Objects*. Project Advanced Web Technologies (Open Distributed Systems) at TU Berlin, SS 2016.
- Quell, Lars; Popescu-Zeletin, Radu (Examiner); Krauss, Christopher (Supervisor). *Digital Video Advertising Displayed on Smart TVs as a Revenue Model*. Seminar Paper at TU Berlin, March 9, 2014. Seminar Paper at TU Berlin, 2014.
- Krischik, Julia; Pfeiffer, Benjamin; Ankert, Madlin; Häuser, Sarah; Ait addi, Abdelkrim; Tramberend, Henrik (Examiner); Krauss, Christopher (Supervisor). *User Centric Data Mining*. Beuth Master Project at Beuth University, WS 2014/2015.
- Deubler, Claudia; Hanke, Hans-Christian; Lehmann, Stephanie; Matuschek, Felix; Schäfer, Claudia; Tramberend, Henrik (Examiner); Krauss, Christopher (Supervisor). *Social Recommendations Through Textual Analysis*. Beuth Master Project at Beuth University, SS 2014.
- Krey, Maya; Örgün, Beste; Maviengin, Ilyas; Wetzler, Rene; Tramberend, Henrik (Examiner); Krauss, Christopher (Supervisor). *HbbTV App Development for Personalized Content Presentation*. Beuth Master Project at Beuth University, WS 2013/2014.
- Fritsch, Igor; Krauss, Christopher (Supervisor); Witzke, Marcus (Supervisor). *Development of DVB-T Probes using Raspberry-Pis*. OKS Project at TU Berlin, SS 2014.
- Jankowski, Pawel; Wierzbicki, Filip; Golenia, Jan-Eike; Krauss, Christopher (Supervisor); Witzke, Marcus (Supervisor). *Trend Predictions through linear Regression*. OKS Project at TU Berlin, SS 2014.
- Guener, Merve; Kamburoglu, Sidika; Krauss, Christopher (Supervisor); Witzke, Marcus (Supervisor). *Semantic Keyword and Sentiment Analysis – Processing of Textual Data for the Prediction of Usage Behavior*. OKS Project at TU Berlin, WS 2013/2014.
- Herrmann, Franco; Jung, Volker; Thome, Karl; Krauss, Christopher (Supervisor); Witzke, Marcus (Supervisor). *Context-aware Interaction Interpolation in a Cross-Device-Game "Frozen Bubbles"*. OKS Project at TU Berlin, SS 2013.
- Sendker, Andrea; Bade, Tassilo; Ukrow, Till; Krauss, Christopher (Supervisor); Witzke, Marcus (Supervisor). *Smart TV Recommendation App "What To Do" based on the Samsung SDK*. OKS Project at TU Berlin, SS 2013.

A.6. Lectures, Presentations, and Talks

- Krauss, Christopher. *How Recommender Systems Leverage Technology Enhanced Learning.* Invited Talk at Holtzbrinck Technology Day, Springer Nature, Berlin, December 6, 2017.
- Krauss, Christopher. *Teaching Advanced Web Technologies with a Mobile Learning Companion Application.* Paper Presentation at The 16th ACM World Conference on Mobile and Contextual Learning (mLearn 2017), ACM, October 30 - November 1, 2017, Larnaca, Cyprus.
- Krauss, Christopher. *Adaptive Learning for Improving Cornelsen's Awarded Clevery App.* Industry-Funded Consulting / Full-Day Workshop at Cornelsen Verlag Berlin, October 19, 2017.
- Krauss, Christopher. *Recommender Systems and How They Support Learners.* Guest lecture in Current Topics in Media Informatics at Beuth University of Applied Sciences Berlin, October 18, 2017.
- Krauss, Christopher. *Personalized Services through Data Mining and Recommendation Engines.* Guest lecture in Current Topics in Media Informatics at Beuth University of Applied Sciences Berlin, April 12, 2017.
- Krauss, Christopher; Dinziol, Martin; Scharp, Michael; An, Troung-Sinh. *Smart-Learning – Medieneinsatz in der handwerklichen Weiterbildung.* Workshop auf der BMBF Fachtagung eQualification 2017, 21. Februar 2017, Berlin, Germany.
- Krauss, Christopher. *Lern-Empfehlungen und -medienkonvergenz.* Eingeladener Impulsvortrag auf dem Arbeitskreistreffen Learning Solutions der Bitkom im Rahmen der Learntec 2017, 26 January 2017, Karlsruhe, Germany.
- Krauss, Christopher. *„Smart Learning": ein Lernempfehlungssystem für digitale Medien.* Learntec 2017, 24 - 26 January 2017, Karlsruhe, Germany.
- Krauss, Christopher; Steglich, Stephan. *Recommendation Engines and Data Mining.* Lecture in Advanced Web Technologies at Technische Universität Berlin, Berlin, December 15, 2016.
- Krauss, Christopher. *Personalized Services through Data Mining and Recommendation Engines.* Guest lecture in Current Topics in Media Informatics at Beuth University of Applied Sciences Berlin, December 14, 2016.
- Krauss, Christopher. *SemA – Semantische Analysen für Lernempfehlungssysteme.* Invited Short-Talk at Holtzbrinck Technology Day, Munich, December 8, 2016.
- Krauss, Christopher; Steglich, Stephan. *TV App Development.* Lecture in Advanced Web Technologies at Technische Universität Berlin, Berlin, November 24, 2016.
- Krauss, Christopher. *Personalized Services through Data Mining and Recommendation Engines.* Guest lecture in Current Topics in Media Informatics at Beuth University of Applied Sciences Berlin, May 25, 2016.

- Krauss, Christopher. *Supervised Speech Act Classification of Messages in German Online Discussions.* Paper Presentation at The 29th AAAI International Florida AI Research Society Conference (FLAIRS-29), AAAI, May 16 - 18 2016, Key Largo, USA.
- Krauss, Christopher. *Smart Learning: Time-dependent Context-aware Learning Object Recommendations.* Poster Presentation at The 29th AAAI International Florida AI Research Society Conference (FLAIRS-29), AAAI, May 16 - 18 2016, Key Largo, USA.
- Krauss, Christopher. *eLmL Session 2: Hybrid learning.* Session Chair at The Eighth International Conference on Mobile, Hybrid, and On-line Learning (eLmL 2016). IARIA, April 24 - 28, 2016 - Venice, Italy.
- Krauss, Christopher. *The Smart Learning Approach - A mobile Learning Companion Application.* Paper Presentation at The Eighth International Conference on Mobile, Hybrid, and On-line Learning (eLmL 2016). IARIA, April 24 - 28, 2016 - Venice, Italy.
- Krauss, Christopher; Steglich, Stephan. *Semantic Web, Data Mining and Recommendation Engines.* Lecture in Advanced Web Technologies at Technische Universität Berlin, Berlin, 7th January 2016.
- Krauss, Christopher; Steglich, Stephan. *TV App Development.* Lecture in Advanced Web Technologies at Technische Universität Berlin, Berlin, 12th November 2015.
- Krauss, Christopher. *Personalized Services through Data Mining and Recommendation Engines.* Guest lecture in Current Topics in Media Informatics at Beuth University of Applied Sciences Berlin, 10th November, 2015.
- Krauss, Christopher; Arbanowski, Stefan. *HbbTV – deep dive and hands-on.* Tutorial at the 5th Media Web Symposium, Berlin, 20th May 2015.
- Krauss, Christopher. *Finding what you are looking for - Recommendation Engines für eine bessere Distribution.* Funded Fullday Workshop at LINKED PRODUCTION WORKSHOPS, Fraunhofer FOKUS, Berlin, 30th April 2015.
- Krauss, Christopher. *Personalized Services through Data Mining and Recommendation Engines.* Guest lecture in Current Topics in Media Informatics at Beuth University of Applied Sciences Berlin, 22th April, 2015.
- Krauss, Christopher. *How HbbTV leverages SmartTVs in Europe.* The First International Workshop for Strengthening China Collaboration on ICT research with Europe (CHOICE) in Immersive and Interactive Media. CHOICE, Beijing, China, 31th March, 2015
- Krauss, Christopher. *Neuartige Möglichkeiten von Werbeschaltungen im digitalen Medienumfeld auf First- und Second Screens.* Invited Talk at the Plenary Meeting of the Verband Privater Rundfunk und Telemedien e.V. (VPRT), Berlin, 27th November 2014.
- Krauss, Christopher; Seeliger, Robert. *No Content without Context – Multimedia Recommendation Engines.* Funded Fullday Workshop at LINKED FILM & TV WORKSHOPS,

Transfer Media, Potsdam, 5th November 2014.

- Krauss, Christopher. *Preference Ontologies based on Social Media for compensating the Cold Start Problem.* The 8th International Workshop on Social Network Mining and Analysis (SNAKDD). ACM, 23th August 2014, New York.
- Krauss, Christopher. *Predictive Data Mining and Recommendation Engines.* Guest lecture in Current Topics in Media Informatics at Beuth University of Applied Sciences Berlin, 2014.
- Krauss, Christopher. *Insights into the development of cutting-edge HbbTV Apps – 10 golden rules for TV App development.* TV Ring Workshop at 4th Media Web Symposium Berlin, 2014.
- Arbanowski, Stefan; Krauss, Christopher. *Multiscreen Development – Video delivery.* Guest lecture in Current Topics in Media Informatics at Beuth University of Applied Sciences Berlin, 2013.
- Arbanowski, Stefan; Krauss, Christopher. *Future Applications and Media @ Beuth University.* Guest lecture in Current Topics in Media Informatics at Beuth University of Applied Sciences Berlin, 2013.
- Krauss, Christopher. *Recommendations as a Service Platform – Intuitiver grafischer Editor zur Erstellung von Empfehlungs-Engines.* IFA TecWatch Forum: New TV-Day at IFA Consumer Electronics, 2013.
- Krauss, Christopher. *TV Predictor: personalized program recommendations to be displayed on SmartTVs.* 2nd International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications (BigMine '13). ACM, New York, NY, USA.
- Arbanowski, Stefan; Krauss, Christopher. *Future Applications and Media @ Beuth University.* In: Guest lecture in Current Topics in Media Informatics at Beuth University of Applied Sciences Berlin, 2013.
- Krauss, Christopher. *Introduction into Data Mining – Recommendations for Interactive Media.* In: Guest lecture in Electronic Commerce of Axel Küpper at Technische Universität Berlin, 2013.
- Arbanowski, Stefan; Krauss, Christopher; George, Lars. *IPTV, Hybrid TV & WebTV – Future Applications and Media @ Beuth University.* In: Guest lecture in Current Topics in Media Informatics at Beuth University of Applied Sciences Berlin, 2012.
- Krauss, Christopher. *OIPF – The importance of Standards for SmartTVs.* In: Apps Hub at the IPTV Worldforum, 2012.

B. Extended Texts

In this Chapter, texts are presented that have not been integrated into the actual dissertation. However, they show exciting aspects or details in particular areas that are worth to be mentioned.

B.1. Realized Adaptive Learning Projects

To understand the technology decisions and the overall research approach, it is essential to know about the history of the Learning-related technologies and the projects that led to specific stages of extension. The related projects are listed and described in this section.

Smart Learning

The starting point to develop a learning infrastructure was the Smart Learning Project[156] which was originally designed for a specialized course at the Chamber of Crafts in Berlin. The project started in October 2014 and ended over three years later in December 2017. Within this project, the general smart learning infrastructure has been developed (including the initial authoring tools, a *Learning Companion Application*, a Learning Content Repository, a *Learning Record Store*, Learning Analytics, an initial version of the *Smart Learning Recommender* as well as a Middleware for data exchange [157]). This infrastructure builds the foundation for the collection of learning data and the integration of educational *Recommender Systems*. The project is extended in a consecutive project that focuses on online training for the craft and services sector. The consecutive project "[SLOW] – Smart Learning for online training" is coordinated by the *Fraunhofer-Institute for Open Communication Systems* (Christopher Krauss).

Software Campus

In March 2015, a second project, namely "[SemA] Machine Learning and Semantic Analyses to improve Predictive Data Mining Algorithms"[158], focused on the design and evaluation of

[156] The Smart Learning project is sponsored by the German Federal Ministry of Education and Research (Bundesministerium fuer Bildung und Forschung – BMBF) under the project funding number 01PD14002A. The four involved project partners are Handwerkskammer Berlin/ Chamber of Crafts Berlin (Coordinator), Fraunhofer FOKUS, Beuth Hochschule für Technik Berlin, Institut für Zukunftsstudien und Technologiebewertung.

[157] The *Learning Companion Application* as user front-end as well as the *Smart Learning Recommender* have been developed primarily by the Fraunhofer FOKUS team led by Christopher Krauss. The Authoring Tools, Middleware, Learning Content Repository and Learning Analytics have been designed by the team of Beuth Hochschule für Technik Berlin. The *Learning Record Store*, Learning Locker, as well as the *LMS*, Moodle, which was needed for the first user management, are open source technologies.

[158] The SemA project is part of the Software Campus program and sponsored by the German Federal Ministry of Education and Research (Bundesministerium fuer Bildung und Forschung – BMBF) under the project funding number 01IS12053. This project had a duration of 2.5 years and ended in August 2017.
Software Campus. See: `http://www.softwarecampus.de/en/home/` (Accessed: 12.06.2017)

algorithms for an educational *Recommender Systems*. The developed algorithms have been iteratively developed, evaluated and optimized to extend the *Smart Learning Recommender*.

FOKUS-Akademie

The *Fraunhofer-Institute for Open Communication Systems* published a learning platform for its own that focuses on the training of expert knowledge in a blended-learning manner. This project builds upon the reference architecture and the expertise of the Smart Learning project. It further introduces components, such as another *LMS*, Open Edx, encapsulated user management for course enrollments and a payment solution for interested learners. With these extensions, the infrastructure enables learners to inquire, pay and enroll for courses by themselves – without the involvement of administrative staff.

Fraunhofer CLM

To merge the learning activities of different Fraunhofer Institutes [159], an initiative started in July 2017 to develop a unique Fraunhofer *LMS* infrastructure based on existing Fraunhofer technologies. The initiative is called Fraunhofer *Common Learning Middleware (CLM)*. The *Fraunhofer-Institute for Open Communication Systems* coordinates the activities[160] and supports the efforts with expertise from and components of the realized Smart Learning project.

Additional Activities

The *FOKUS* expertise is additionally inquired by industry partners for workshops and talks (e.g., Cornelsen, Holtzbrinck) and by the public sector (e.g., Behörde für Schule und Berufsbildung – BSB Hamburg). Besides the funded project activities, especially the teaching assignments of the partners allowed to optimize and evaluate the developed components. The Beuth University[161] used the system to organize an online-only course about the Framework Java-FX. Christopher Krauss, in turn, held guest lectures at the *Technische Universität Berlin* (and additionally at Beuth University) which enables the team to test and evaluate the learning infrastructure in other contexts – for teaching computer scientists in a course named *Advanced Web Technologies*. Moreover, this educational activity further allowed for supervising related Bachelor and Master theses as well as student projects in the area of *Recommender System* and *Technology Enhanced Learning*.

B.2. Formatting Styles

Sub-Headline

This section briefly introduces the citation, reference and formatting styles used in this dissertation. The document uses four hierarchy level for headlines, namely chapter (1.), section (1.1), subsection (1.1.1) and only in some cases paragraphs (1.1.1.1). As customary, these headlines are shown at the beginning of the text as well as in the *Table of Contents*. They represent a classification of the main topics introduced in the following text block. However, as an issue might be discussed over several pages, Sub-Headlines are shown right next to the beginning of

[159] Namely the Fraunhofer Zentrale, represented by the Fraunhofer Academy, Fraunhofer FIT, Fraunhofer FOKUS, Fraunhofer IDMT, Fraunhofer IOSB, Fraunhofer IAIS and Fraunhofer IML

[160] Christopher Krauss coordinates the project together with Eva Poxleitner from Fraunhofer Academy.

[161] Represented by Prof. Dr. Agathe Merceron

a paragraph and titles the current idea. To avoid unneeded complexity, these sub-headlines are not shown in the *Table of Contents*, because they mainly assist the reader's orientation. See an example at the beginning of this paragraph.

Abbr.

This thesis discusses a variety of technical terms. Especially longer terms, e.g., consisting of more than just one word, are abbreviated for better readability. Thereby, new technical terms, such as *Recommender System (RS)*, are introduced first as long and afterward (in brackets) as short term. Subsequent, either the short term *RS* or the long term *Recommender System* can be used. All abbreviations show a particular formatting style as they can be found in the *List of Abbreviations* at the beginning.

References

In some cases, only online references show the current state of the art as there is no print media available for that specific topic. Online references are given in footnotes with the title, the *Uniform Resource Locator (URL)*, and the access date. Moreover, in between the dissertation text, some short quotes are given from famous persons on the current topic:

"The power of the Web is in its universality.
Access by everyone regardless of disability is an essential aspect."
Tim Berners-Lee, W3C Director and inventor of the *World Wide Web (WWW)* [162]

Self-Citations

Besides externally cited work, the author already presented parts of this research in international industrial and academic conferences. The associated papers were published in proceedings or journals and present research in the areas concerned. A self-reference presents these results and appears in a special way:

> "This is the formatting style of a peer-reviewed and published text of the author! [...] Each text is longer than three lines and should not exceed a whole page. Citations within a self-reference are formatted according to the citation format of this dissertation. The reference to the original paper is given below the citation box. In a footnote, a small indication of the co-authors and main contributors to that excerpt are listed as well as a brief introduction describing the context if needed"
>
> Cf. [Reference, Page(s)][163].

There are several reasons for presenting self-references in a dissertation. First and foremost, a published text was already peer-reviewed, usually blind (without knowledge of the author names), by at least two, sometimes up to six, independent experts. Thus, such a paper was already judged concerning composition, related work, novelty, significance and other academic aspects. Especially

[162] See "edX W3C course": `https://courses.edx.org/courses/course-v1:W3Cx+HTML5.1x+3T2016/info` (Accessed 01.12.2016)

[163] This is not a real self-citation, but a short text presenting its format. This text was exclusively written by the author of the dissertation.

the date of publication indicates its timely relevance – that the content might be released early when some of these topics are trending. Each publication focuses on a separate, self-contained topic with its literature review, concept, evaluation and result discussion. Excerpts presented in this work will focus only on significant parts and so bring additional value to the underlying textual description. Of course, a self-citation also helps to avoid redundant writings about topics that are already published and indirectly, avoids self-plagiarism, as well.

B.3. Learning Trends

Learning Trends

Technology Enhanced Learning comes in different flavors. Besides access to *LOs*, new topics develop continuously. Figure B.1 shows a rating of learning experts for different technological directions. The participants of this survey were asked to rate each trend on a scale from 0 (unimportant) to 5 (important).

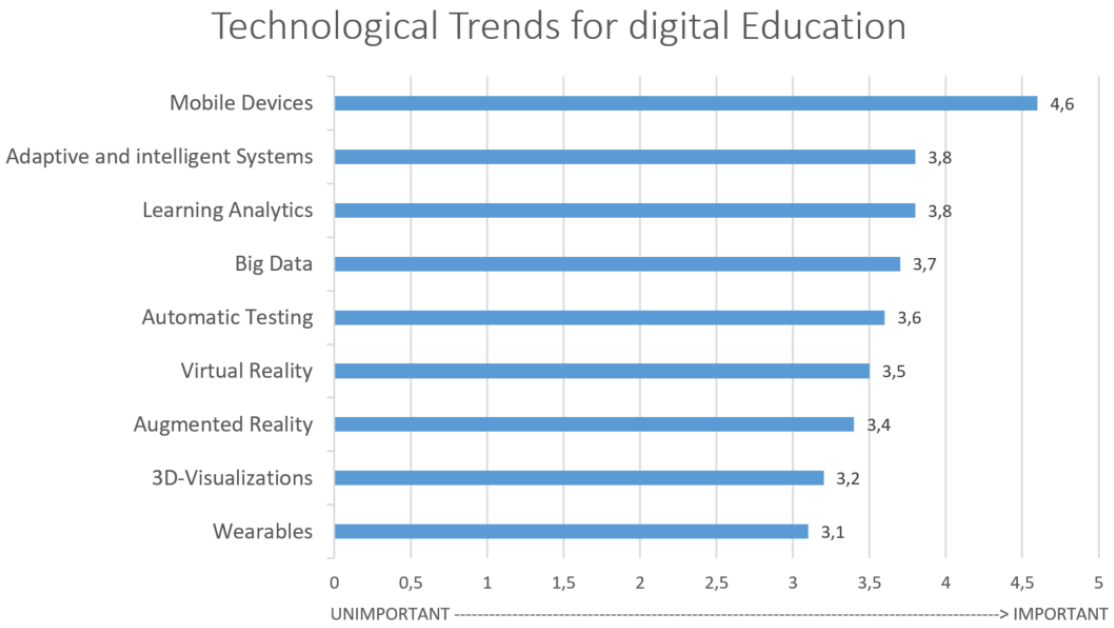

Figure B.1.: 57-66 experts were asked about the importance of different technological trends for digital education [195].

Mobile Learning

As can be seen, learning on mobile devices seems to be a critical aspect of future developments in the area of *TEL*. This finding corresponds to the need for anywhere and anytime learning, a recent trend of the bustling target group, as most learners try to organize their learning periods whenever they have free time.

Adaptive Learning

Place two up to place four of the most important technological trends can be classified as topics requiring data analysis. Adaptive and intelligent systems, among others with the help of

Recommender Systems, are expected to bring enormous potentials for all: learners, content creators, and teachers. *Intelligent Learning Technologies (ILT)* and *Intelligent Tutoring Systems (ITS)*, for instance, bring intelligence into the digital learning environment and can be seen as a subclass of *TEL* systems. Learners might organize their learning more efficient and more effective. This dissertation will focus on *Intelligent Learning Technologies*, especially for learners. Content creators, in turn, can benefit from digital assistance during the media creation and compilation. Last but not least, teachers get hints for general weaknesses and learning topics to focus on in depth. The latter also interferes with Learning Analytics which also aims at analyzing the participants' studying patterns to conclude helpful information for the teacher.

Further TEL Trends

The basis for these prediction tasks is, of course, data. The more data is collected, the more meaningful are the predictions. Big data stands for the collection and processing of huge information sets of detailed usage and activity data. Automatic Testing, Virtual Reality, Augmented Reality, 3D as well as Wearables (the less important trends in Figure B.1) play a minor role for this dissertation but might be very useful for additional content presentations.

B.4. Computer-assisted Educational Tasks

Darabi et al. [78] identified the 54 most important tasks which educators of distance education settings typically fulfill. Thereby, almost all can be taken over or at least be supported by adaptive technologies. The following list shows an order of the ten most important tasks that have been ranked by 148 online-course instructors (cf. [78, p. 111]). It also shows how these tasks can be supported by software components[164]:

1) "Review the course for accuracy": This can be supported by Learning Analytics which collect and process the learners' learning activities automatically and give hints for the instructional staff.
2) "Share supplemental learning resources with the learners": Giving hints and recommendations for additional and appropriate learning items is a typical task of a *Recommender System* which, especially at a particularly course scale, can be done more effectively and better personalized by algorithms due to the fact that it is expensive for educators to keep track of all students individually.
3) "Assess learners' attainment of learning objectives": The log of learning activities can be analyzed by Learning Analytics components to give automatic feedback for both, learners and instructors.

4, 5) "Maintain expertise in subject areas [... and] in online instructional techniques": While this is still the most important task a human instructor needs to solve, the instructor

[164]The ideas, how these tasks can be supported by software components, comes from the author of this work.

can be assisted by semantic search engines and *Recommender Systems* in order to identify appropriate resources which help to maintain this knowledge.

6) "Maintain record-keeping related to course activities": Timetables with lecture dates, assessment deadlines and exams can be managed once, and the required learning activities can be observed automatically or at least semi-automated by analyzing the learners' activities.
7) "Make changes as needed to maintain course accuracy": While the course should be adjusted and optimized by educational staff, the hints for possible weaknesses and amendable content can come from software components that process course activities and learning assessments.
8) "Use strategies to create a friendly and open environment": The perception of a course can be influenced by the participating learners and instructors and by the technical environment. While interpersonal factors cannot be directly affected, the technical environment can adapt to the users' needs. That can be done by offering settings for adjusting the look and feel of the platform according to the users' needs, by providing helpful information in different situations or by giving access to appropriate components, such as different internal and external learning media, personal progress overviews and communication tools.
9) "Use questions that promote higher order thinking": This seems to be the most challenging task for a computer and is still mostly solved by human instructors. However, researchers keep improving chatbots which might be eligible to take over this task in the future.
10) "Provide feedback on the accuracy of learners' statements": The self-reflection of learners can be requested by a system at particular points in time and be automatically checked for accordance with a predicted knowledge level which might give a clue on its accuracy.

B.5. Brief History of Recommender Systems

Paul Resnick and Hal R. Varian published an often-cited survey of *Recommender Systems* in the end 90's [226]. They state the first recommender was "Tapestry" in 1992. Goldberg et al. developed this mail filtering system at the Xerox Palo Alto Research Center that selects "interesting documents no matter what list they are in [...] by involving humans in the filtering process" [113, p. 61]. While Goldberg et al. named their approach *Collaborative Filtering (CF)*, Resnick and Varian introduced the more general term *Recommender System* for this kind of intelligent filtering. This mail filtering system gave the starting signal for more research in the area of personalized filtering: In 1994, Resnick et al. introduced GroupLens – an open architecture for *CF* of Netnews articles [225]. One year later, Hill et al. proofed the appropriateness of "virtual community recommendations" to select other types of media items [127, p. 194]. They started with a set of 500 videos but noted that such an approach is also applicable to other domains, such as "music, books and catalog products" [127, p. 201]. Later in 1995, an approach was introduced "for making personalized recommendations from any type of database to a user based

on similarities between the interest profile of that user and those of other users" [243, p. 210]. Henceforward, the *RS* domain got increasing attention in numerous sub-areas, first and foremost, driven by the internet industry that realized its commercial potential.

CbF Techniques

Content-based Filtering (CbF) techniques are older than the term *Recommender System*, as *CbF* describes a set of methods that try to match the items attributes with the user's interest towards the items attributes [9]. The first mail filters used simple *CbF* techniques before ratings were introduced [113]. To "recommend movies to user c, the content-based recommender system tries to understand the commonalities among the movies user c has rated highly in the past (specific actors, directors, genres, subject matter, etc.). Then, only the movies that have a high degree of similarity to whatever the user's preferences are would be recommended" [9, p. 735-736].

First CF

Collaborative Filtering (CF) represents the most common Filtering approach as it takes the interests of other users into account. While *Content-based Filtering* was frequently used, even before the first *RS* was introduced, Tapestry (the mail recommender in 1992) used the reactions of other users, to filter a huge amount of documents. Unintentionally, the developers of Tapestry, Goldberg et al., gave the first definition for *Collaborative Filtering*: "Collaborative filtering simply means that people collaborate to help one another perform filtering by recording their reactions to documents they read" [113].

B.6. Typical (TEL) Recommender Tasks

There are a lot of different definitions of *Recommender System*s. However, one could also describe a recommender by the most common operations. According to Herlocker et al. [125] typical tasks of *Recommender Engines* are:

1. Annotation in Context: Items are classified whether they are appropriate in different contexts for specific users.
2. Find Good Items: Besides the classification of item appropriateness, items are rated and ranked by considering historical consumption data.
3. Find All Good Items: In some cases, it is important to present not only some good items but a complete list of these items. Therefore, service providers need not ensure that the false negative rate is negligibly low.
4. Recommend Sequences: The prediction of an item sequence is different from just ranking the most appropriate items, as the relevance of a sequence item is directly affected by the previous item: e.g., when recommending an order of songs.
5. Just Browsing: Some users have no clear motivation for the usage of a web service, but want to be inspired by recommendations. Therefore, the accuracy of item predictions plays a minor role, while the presentation is more important for the overall user acceptance.

6. Find Credible Recommender: Some users interact with recommender systems just to evaluate the effect of changes in their user profile. Especially, when new *RS*s were introduced, users play around with these systems to check their appropriateness and to create trust in it.

From a user perspective, Herlocker et al. [125] describe some recommender tasks that are a direct result of the user's motivation:

7. Improve Profile: Users rate items because they believe in improving their user profile to get better recommendations.
8. Express Self: Some users, in contrast, do not primarily want to get good recommendations for themselves, but want to express their opinion on items.
9. Help Others: Some express their opinion to contribute to a community actively which is not necessarily the same as self-expressing as mentioned above.
10. Influence Others: In contrast to the two previous tasks, a low percentage of users want to actively influence recommendations of other users to play the role of an adviser.

In addition to that, Manouselis et al. [191] defined the following tasks – especially for *TEL* playing a key role in this dissertation – that could also fit for generic *Recommender System* and thus, need to be added here:

11. Find Novel Resources: Especially very active users know most of the presented items already. Moreover, some services, such as news websites, need to present current items. Therefore, recommenders may present only recent appropriate items.
12. Find Peers: In contrast to finding good items, recommender systems may also support in finding similar users, such as nearest neighbors or complementary peers.
13. Find Good Pathways: In some cases, recommended sequences don't represent the best order for the demanding user or situation. That is why the presentation of alternative sequences can be of interest – e.g., alternative learning paths through a set of learning resources or through course contents.

From a user perspective, a *RS* should effectively support their needs by performing the previously mentioned tasks and goals very well. Thereby, the interests and needs of the different players (e.g., user and service provider) need to be balanced very well to offer valuable services for both [146].

Filtering Steps

As the most common operation of a *RS* is to aggregate a sub-list of most relevant items out of the fully available item list, the technical task is called filtering. Although, there are many different approaches to receive the final Top-N item list, in most cases, the starting point is the complete item data set. Thereby, the calculation process can consist of one or multiple steps like the following:

1. Item selection via white- or blacklists: On a whitelist, every item that meets pre-defined criteria, will stay on the resulting list. In contrast, on a blacklist, every item that does not meet pre-defined criteria, will be excluded from the resulting list.
2. Item categorization: Gives a label to an item to group multiple elements by similar attribute values.
3. Item scoring: Determines a score that represents the item's appropriateness.
4. Item set sorting: The list will be sorted by pre-defined criteria, e.g., its determined score.
5. Item set limitation: The resulting list will be limited only to the first n items of the sorted order or by using a threshold for the determined score – this step is similar to the selection process at the beginning.

Of course, there can also be a loop of these tasks, to iteratively shrink the resulting list by applying different approaches.

B.7. Recommender Systems from a Business Perspective

Business Perspective

From a business point of view, there are many reasons why service providers should use *Recommender Systems* [146]: They increase the number of sold products, as more items are discovered that fit the users' needs. Moreover, *RS*s support to sell more diverse products, since it could shift the user's attention from mainstream products to long-tail items. Additionally, users are more satisfied, because the item selection process is faster and more efficient. When recommendations are relevant and engaging, the user experience improves. And the better a service adapts to a user, the more likely the user comes back. As a consequence, more users will actively try the services. And finally, the bigger the community of a *Recommender System* is, the better is the understanding of the customers' needs which leads to more accurate recommendations and to more sold products.

Amazon attributes 35% more sales to a *Recommender System* [165] – which, by the way, is controversial, because it is still not directly measurable. However, it shows the expected value of this approach for the e-commerce sector and the benefits for service providers.

B.8. User and Item Relations in Recommender Systems

A *Recommender Engine* requires data about the community and the items to be recommended. This information is called metadata and describes aspects of an object (here a user or an item) in a machine-readable way. Users and items, in turn, need to be related to fit the individuals' needs.

[165] See "Aggregate Knowledge raises USD 5M from Kleiner, on a roll": `http://venturebeat.com/2006/12/10/aggregate-knowledge-raises-5m-from-kleiner-on-a-roll/` (Accessed: 07.09.2016)

Figure B.2 illustrates the central relationships between users and items, in terms of feedback, the similarity between users and users as well as the similarity between items and items.

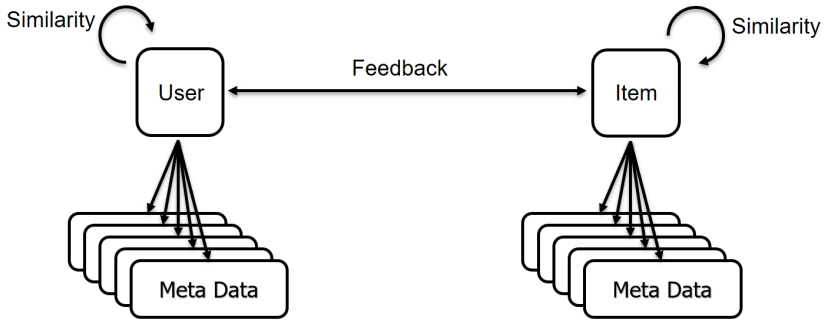

Figure B.2.: Relationship between users and items in Recommender Systems

User Feedback

The five stars scale seems to be the most common feedback type in the web to relate users and items. Among others it is used by Amazon[166] and Netflix[167]. Indirectly, the five stars scale refers to the Likert-scale that is a common "Technique for the Measurement of Attitudes" and tries to map human opinions to a numerical scale, here to the five-point statements "strongly approve, approve, undecided, disapprove, and strongly disapprove" [179, p. 15]. However, a lot of web services show very creative alternatives: "likes" as known from Facebook[168], "views" as known from Youtube[169], 10-Rating points as used by IMDb[170] or even the TV Predictor in Movisto developed by Fraunhofer FOKUS Cf. [167]. Besides the so-called explicit feedback, where a user provides feedback on purpose, there is another feedback type: implicit feedback is collected without being noticed by the user [146, 270, 57]. For instance, when visiting a product-page of Amazon or when watching a video on Youtube. The TV Predictor uses both feedback types, as well:

> "In order to calculate recommendations the TV Predictor must know about the users interests. Therefore the engine differs between two different feedback types [...]:
>
> - Automatically tracked watch behavior: The client sends in regular intervals messages indicating the watched channel, so the server can lookup for the current playing program in the database.
> - Manually given ratings: The user can explicitly provide ratings for single programs in range of [1, 10]"
>
> Cf. [167, p. 66][171].

[166] See Amazon: `www.amazon.com`
[167] See Netflix: `www.netflix.com`
[168] See Facebook: `www.facebook.com`
[169] See Youtube: `www.youtube.com`
[170] See IMDb: `www.imdb.com`

In addition to explicit and implicit data, Verbert et al. [270, p. 321] listed a third category of user feedback – "inferred data". This data can be extracted by further processing explicit and implicit data and matching it with additional parameters, for instance "to estimate the current task of the user".

B.9. Additional Recommender Approaches

Besides classical *Content-based Filtering* and *Collaborative Filtering*, a lot of other approaches exist. Mostly, there is neither a common definition nor a common classification for these approaches, because their concepts may fall into multiple filtering classes. That is why the following approaches are mentioned without a binding classification.

Machine Learning

A lot of recommender systems use *Machine Learning (ML)* techniques, such as Clustering, instead of classical Filtering approaches. The main difference is that *ML* techniques are designed to perform general regression or classification tasks that do not need to cover recommender aspects necessarily. However, these online and offline *Machine Learning* algorithms can perform similar functions as classical filtering approaches. For that reason, some of them are introduced as filtering techniques. Another way of classifying *ML* approaches is in the form of learning: In supervised learning, a human supervisor needs to label training data, so that an algorithm can learn the features associated with that label and predict the labels of future items. Unsupervised learning, instead, tries to find patterns and structures of unlabeled data without any human confirmation [146, p. 48].

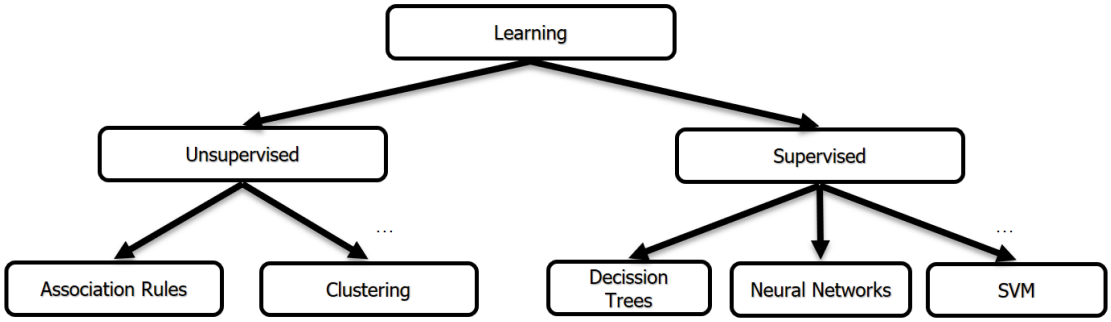

Figure B.3.: Machine Learning approaches for recommender systems

Park et al. [209] also classified often used data mining techniques for *Recommender Systems* in the research literature. In 2012, they published the details on the analysis of 210 scientific papers in 46 journals. The explored papers can be clustered into eight significant machine learning approaches: Association Rules, Clustering, Decision Trees, k-Nearest Neighbor, Neural Networks,

[171] This extract was written by Christopher Krauss. The TV Predictor was developed first as showcase for Bertelsmann Arvato to be displayed on SmartTVs (shown at IFA 2012) and afterward as website component "Movisto" for `www.rtv.de` (Dec 2012 until June 2013).

Link Analysis, Regression and other heuristic methods.

Preference Filtering

Machine Learning can also be used for *Preference Filterings (PFs)* that tries to map the preferences of a user to the attributes of an item – so for instance in the TV Predictor recommendation engine using a *Support Vector Machine (SVM)* as a binary classifier:

> "Its goals are to find items that fit the users' preferences. These preferences originate from manual user input or by learning them automatically. Therefore user interests (represented as attributes) are mapped to the item attributes, such as a sports interest of a user is mapped according to the category of an item (e.g. a sports program) (cf. [299, p. 244]). Support Vector Machines (SVMs) create hyperplanes that divide specific space with according points into two spaces. Each point represents an item, and the according coordinates are its attributes. The Support Vector Machine will find the best fitting hyperplane that is the borderline between two classes (spaces). This can be done very effectively in multidimensional spaces. (cf. [130, p. 147], [289, p. 401-402])"
>
> Cf. [167, p. 68][172].

In the *RS* of the TV Predictor, each user trains his/ her own *SVM* with his/ her rated programs respectively their features. For each user, a threshold is calculated, that splits high and low ratings into equal parts. This threshold acts as the hyperplane, and the result is a prediction whether an item would be rated good or bad.

> "The input data (the training as well as the prediction data set) are slot metadata of a watched program that must be converted into a list of numbers. So it is only applicable to use features that can be mapped to the required numeric range of [-1,1]:
>
> - Slot time: The weekday (0-6), the duration of the slot (0-240 min) as well as the begin and end time (0-24 hours) are used.
> - Program data: The production year (1920-2020) and the country String is used. The usage of Strings requires a mapping to more than one feature. So there is a feature for each possible country (D, GB, USA, E, F etc.) with the value 1 (it is this country) or 0 (it is not).
> - Channel data: The same procedure as for the countries applies to the main 22 channels.
> - Genre data: Same as above for 69 genres.
> - Category data: Same as above for 6 categories.
>
> So at all, there are 109 features for each slot. To get a prediction, each training slot has to be labeled. In this case, the label is numeric and either a 1, when the rating of the current user for that slot is higher than 7.5, or apart from that, it is a -1"

[172] This paper has primarily been written by Christopher Krauss.

Cf. [167, p. 68][173].

In an experiment, nine participants provided between 15 and 113 ratings on different TV programs – 28 on average. To evaluate the accuracy of the *SVM*, a 10-fold cross-validation was utilized on previously rated items. The accuracy ranges between 60 and 96 % for the different nine users – 85 % on average.

Utility-based Filtering

In the style of *Knowledge-based Filtering*, Burke sees *Utility-based Filtering* as special recommender class. Thereby, the user profile is treated as a complex "utility function that the system has derived for the user, and the system employs constraint satisfaction techniques to locate the best match" [52, p. 3]. For that reason, *Utility-based Filtering* can also consider non-product features "such as vendor reliability and product availability" to calculate better context-aware recommendations. This approach, for instance, avoids the recommendation of items with shipping delays when the user's need for that product is urgent. In contrast to Burke's interpretation, this approach is classified as sub-class of *Content-based Filtering* within this work – similar to Item-to-Item Similarity Mapping as defined in [181].

Demographic Filtering

Demographic Filtering (DF) can be used either to enrich the user profile with additional (demographic) data or to add to the existing item attributes some target group specific demographics. Manouselis et al. [191, p. 398] classifies Stereotypes or demographics filtering in *Technology Enhanced Learning* as *Collaborative Filtering* approach. The processed "information can be used to identify the types of users that like a certain object" [212, p. 400] and subsequently, to create "categories of users having similar demographic characteristics" [14, p. 873]. This approach aims at identifying first "which category [a new user] belongs and then [at] applying the aggregate buying preferences of previous users in that category" [14, p. 873]. Thus, *DF* might improve the accuracy and performance of predictions in both algorithm classes: *Content-based Filtering* and *Collaborative Filtering*. Thus, it can be handled as a *Hybrid Filtering* approach.

Community-based Filtering

While the topic of *Recommender System* and different data mining approaches was exhaustively analyzed over the last decades (cf. [119, 11, 145, 38, 37]), the topics of data mining for social media (cf. [230]) and social network analysis [43, 79] play a substantial role in the last few years, because distributed and big data processing as well as graph-based analysis are more and more of interest.

Community-based algorithms try to improve data prediction by taking the social connections into account (cf. [176, 294]). These data can be taken from external social networks (cf. [43, 63]) or own communities (cf. [198]).

SNA allows for re-using existing information to enrich the community data of a closed corpus recommender system. The significant advantage comes from the circumstance that users do not want to enter the same information in multiple web services, but service providers can easily access

[173] This paper presents the results of the Master thesis of Christopher Krauss and has primarily been written by him.

existing data via open *API*s. Facebook, for instance, offers an *API* to access entered likes and past comments. This is used to enrich community data in our TV Predictor *Recommender System*.

> "Our application prototype is a web-based personal Electronic Program Guide (EPG) which is able to offer TV recommendations just seconds after users log in for the first time with their Facebook accounts by parsing their personal data. The program data is being delivered by a recommendation engine called TV Predictor [167]. Moreover, the novel recommendations based on the users Facebook profile are highlighted, and it is even possible to see what the user's friends could like. [...] Basically we want to find out how many of the determined criteria/attributes are matching a certain program. Since metadata like title or genre might not always be written the same way throughout Facebook movie pages and various EPG data provider, we use the Levenshtein distance to improve robustness while matching these strings that is "The smallest number of insertions, deletions, and substitutions required to change one string or tree into another" [216]. We calculate a similarity degree which is 1 minus the Levenshtein *distance* divided by the length ($maxLength$) of the longest of the two strings $S1$ and $S2$:
>
> $$sim(S1, S2) = \frac{1 - distance}{maxLength(S1, S2)} \tag{B.1}$$
>
> [...] The prototype indicates that using textual analysis of Social Networks could compensate the cold start and sparsity problems predicting automatically the interest in otherwise unknown items by 1.8% up to 15.1%. As a result, users can start with a personalized service without giving explicit feedback and by only logging into a Social Network"
>
> Cf. [164, p. 67][174].

This section introduced *Machine Learning* as well as some less important or niche filtering approaches which, moreover, might be classified in different ways.

B.10. An Example of Hybrid Filtering

A representative example of *Hybrid Filtering* is the TV Predictor [167] introducing a novel recommendation engine that shows program recommendations directly on a SmartTV:

> "The recommendation system uses a set of different criteria to make recommendations which correspond to the users' viewing behavior. When users watch TV Predictor enabled channels, they can open the recommendation menu by pressing the according button on their remote control. A set of the best and most relevant programs for the current user will

[174] This paper has been primarily written by Christopher Krauss.

be shown. These personalized recommendations are based on the automatically tracked viewing behavior and explicitly defined program ratings of the registered user or - in case they did not sign up – they will get averaged or well-selected recommendations"

Cf. [167, p. 63-64] [175].

The TV Predictor introduces an example application area as well as some common collaborative and content-based filtering techniques for predicting the users individual watching behavior. As a result, the Top-7 most appropriate TV shows are presented in different categories to the user:

"In order to generate the best and most accurate recommendations, the recommendation system combines the best fitting algorithms in a hybrid way. The usage of these algorithms depends on the user's request:

- Find similar programs to the selected one by using common content-based filtering algorithms, such as the Cosine Similarity, and by using unsupervised learning algorithms, such as Association Rules
- Get program highlights for a specific time period based on the favorite programs of similar users (Pearson Correlation Coefficient) and predictions of program ratings (Slope One)
- Calculate a personalized program guide changing the channel automatically by using Clustering to pre-select programs best fitting the user's interests and rating predictions
- Overlay upcoming program recommendations while watching TV based on recognized behavior patterns (calculated by a Support Vector Machine) to find user interests, such as genres and categories, favored actors, directors and producers or even the preferred channels, weekdays or times to watch specific content"

Cf. [167, p. 64] [176].

As shown in the paper extract, the TV Predictor uses, among others, *Content-based Filtering* in two different flavors. The first one is a classical item-to-item-comparison: When a user watched one movie, other movies that are similar are recommended. The second way of *CbF* is to suggest new movies by matching the user's preferences (metadata of watched films collected during past transactions) with other item's metadata. The result is a list of new movie recommendations that fit the users past watched movies.

[175] The work at the TV Predictor engine started with the master thesis of Christopher Krauss in 2011 – supervised by Prof. Dr. Agathe Merceron. The publication is the result of an extension of this work under the project lead of Mr. Krauss at the *FOKUS*. Arvato RTV (content provider and subsidiary company of Bertelsmann) used the TV Predictor for about one year under the name Movisto and by Media Broadcast (DVB-T/ DVB-T2 network provider in Germany) in the service multithek. The here shown section was written exclusively by Christopher Krauss.

[176] The presented section was written exclusively by Christopher Krauss.

The TV Predictor combines the introduced *Content-based Filtering*, *Collaborative Filtering* and *Preference Filtering* approaches in a hybrid way, as described here:

> "The TV Predictors recommendation engine uses the benefits of Content-based, Collaborative and Preference Filtering as well as [...] offline learning algorithms. [...] The starting point for a typical prediction is the set of all relevant items for the specific situation (e.g. a group of channels, genres or a specific time period). The system analyses all items and existing user-item-relations. If such a relation exists, the engine skips the rating prediction. Otherwise, the introduced algorithms consecutively calculate predictions for the current item. If the results fit into a specific range (so considered as good proposals), the results are merged in a weighted way. If not, the result set is decreased by this item"
>
> Cf. [167, p. 68][177].

B.11. Explanation of the User-Item-Matrix

User-Item-Matrix

For a better visualization and to be prepared for future calculations, the feedback of a user for an item is stored in a user-item-matrix [180]. This matrix consists of columns for items and lines for users. Each cell relates an item to a user and contains a value representing the feedback. Figure B.4 shows an instance of a user-item-matrix with feedback on a one to five stars scale – the more stars, the higher the preference. The service of this example offers five items $i_1, ..., i_5$ and has five customers $u_1, ..., u_5$.

Feedback Prediction

In most cases, only a meager percentage of cells are filled. The empty cells indicate a missing knowledge due to a lack of user feedback on these items. This issue results in multiple challenges, that might bring a lot of trouble to *Recommender Systems*: for instance when there is too little data available for a prediction. However, *Recommender Systems* aim at predicting these missing values by matching the feedback of others. In the example in Figure B.5, the preference of u_1 for i_4 is predicted by analyzing other users in the user-item-matrix.

[177] The TV Predictor engine, developed by Christopher Krauss, predicts the Top-N item for each user in particular situations and uses a hybrid approach consisting of the following approaches: Content-based Filtering with the Cosine-based Similarity, Item-based Collaborative Filtering with Slope One, User-based Collaborative Filtering with Pearson Correlation Coefficient, Neighborhood-based Filter using K-Means Clustering, Rule-based Recommendations with Association Rules utilizing the Apriori Approach and Preference Filtering with the help of a Support Vector Machine for each user.

All algorithms are combined in a cascading, switching or weighted way as mentioned previously. Moreover, an evaluation was conducted to analyze the overall performance as well as the average accuracy.

	i_1	i_2	i_3	i_4	i_5
u_1	5		2	?	3
u_2	5		1	5	
u_3		4		4	4
u_4	4	2	5	3	
u_5	3		1		

Figure B.4.: Instance of a user-item-matrix with five users and five items.

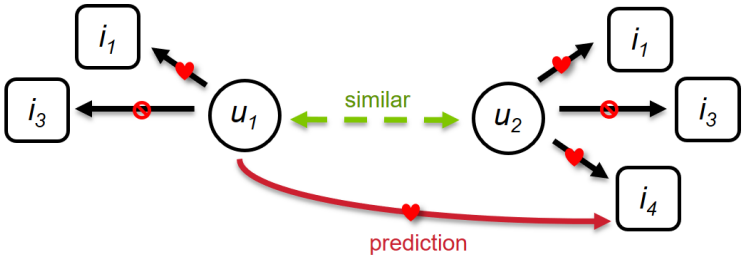

Figure B.5.: Example of Collaborative Filtering based on the introduced user-item-matrix.

B.12. TV Program Association Rules

The Association Rules algorithm consists of two parts. Part one generates frequent itemsets containing these items, which have frequently been consumed together – e.g., by the same user on the same day.

"Therefore, the watch-feedbacks of each user are represented as a transaction. It is also possible to use other user-item-relations, such as the best-rated items (analyzed with the help of a rating threshold) for each user, or the watched items for each user and day. The threshold of total watched time (for genres, programs, etc.) is 20 minutes for the prototype. When using the frequent itemsets for more than only 300 users, the threshold has to adjust dynamically to a sound value. So, the frequent item sets are found for the following domain data objects (the given minimum support values work fine for the test data set):

- Categories (minSupp: 10%)
- Channels (minSupp: 20%)
- Crew Members (minSupp: 11%)

- Genres (minSupp: 10%)
- Programs (minSupp: 6%)
- Series (minSupp: 6%)"

Cf. [167, p. 67-68][178].

The second step generates the association rules for the calculated frequent itemsets and ranks them by their confidence value.

"Some thresholds are needed to adjust this algorithm. Especially the support value ($supp$ in percent or sometimes as absolute value) is necessary and defines the frequency of an item set.

$$supp(X \implies Y) = \frac{supp(X \cup Y)}{|T|} \geq minsupp \tag{B.2}$$

$|T|$ is the total number of transactions, so $supp(X \implies Y)$ is the fraction of all transactions that contain X and Y. The confidence value expresses the conditional probability of Y knowing X and is defined as:

$$conf(X \implies Y) = \frac{supp(X \cup Y)}{supp(X)} \geq minconf \tag{B.3}$$

The minimum confidence value ($minconf$) and the minimum support value ($minsupp$) are the according thresholds, so the goal is to find rules with a support value equal to or greater than $minsupp$ and a confidence value equal to or greater than $minconf$"

Cf. [167, p. 67-68][179].

The results of this algorithm have been evaluated in an online experiment with 40 volunteers. The participants were asked to document every watched program during a three weeks duration in an easy to use web interface. As a result, they viewed 522 different programs (most of them were series and movies) on 36 channels. In total, the summed watch duration result in 311 TV hours.

"All results are based on the values: minimum Confidence (0.5) minimum Lift (1.1) and minimum Cosine (0.66). Over 500 interesting association rules resulted. Some of the strongest rules are stated below:

- Genre: $'Docu - Soap', 'Late - Night - Show' \implies 'Soccer'$,
- Category: $'Entertainment', 'Series', 'Other' \implies 'Report', 'Sport'$,
- Series: $'TheSimpsons', 'Scrubs' \implies 'HowIMetYourMother'$"

Cf. [167, p. 67-68][180].

[178] The here introduced experiment was written and conducted by the author.
[179] The paper extraction was written by the author and reflect the work of [180], [182] and [118].

Below is a list with more results of that experiment:

- Channels:
  - Rule: "ProSieben", "Sat.1" $\Longrightarrow$ "VOX",
  - Rule: "ProSieben", "VOX" $\Longrightarrow$ "Das Erste"
- Programs:
  - Rule: "Die purpurnen Flüsse", "Hilfe, mein Mann ist ein Heimwerker!" $\Longrightarrow$ "Die Küchenchefs",
  - Rule: "Illuminati", "Wer wird Millionär?" $\Longrightarrow$ "Gute Zeiten, schlechte Zeiten"
- Series:
  - Rule: "Freitag, der 13.", "Das perfekte Dinner" $\Longrightarrow$ "Fußball",
  - Rule: "Die Simpsons", "Das perfekte Dinner", "Sportschau live" $\Longrightarrow$ "mieten, kaufen, wohnen",
  - Rule: "How I Met Your Mother", "Die Simpsons", "Scrubs - Die Anfänger" $\Longrightarrow$ "Galileo"
- Crew/ Actors
  - Rule: "Eva Mona Rodekirchen", "Seyhan Derin" $\Longrightarrow$ "Daniel Fehlow",
  - Rule: "Daniel Fehlow", "André Siebert" $\Longrightarrow$ "Seyhan Derin",
  - Rule: "Seyhan Derin", "Daniel Fehlow", "André Siebert" $\Longrightarrow$ "Eva Mona Rodekirchen"

Amazon's Item-to-Item Filtering approach, which produces similar results as Association Rules, belongs to the class of *Item-based Collaborative Filtering* and uses the Cosine-based Similarity measure with ratings instead of attributes. The algorithm searches for similar user ratings based on the ratings the current item has received [180, p. 79]. However, Amazon uses only an excerpt of the whole data set for each request as of the vast amount of existing feedback data and the resulting vector dimensions.

B.13. Pearson Correlation Coefficient

Pearson Correlation Coefficient

"The Pearson correlation coefficient for instance is often used to calculate the similarity of users by considering the items both users rated.

$$pSim(u_1, u_2) = \frac{\sum_{i \in I_{u_1 u_2}} (r_{i,u_1} - \overline{r}_{u_1})(r_{i,u_2} - \overline{r}_{u_2})}{\sqrt{\sum_{i \in I_{u_1 u_2}} (r_{i,u_1} - \overline{r}_{u_1})^2 (r_{i,u_2} - \overline{r}_{u_2})^2}} \tag{B.4}$$

[180] The here introduced experiment was written and conducted by Christopher Krauss.

r_{i,u_1} is the rating of user u_1 for the item i of item set $I_{u_1u_2}$ (a set with existing ratings of both users for each item). $\overline{r}_{u_2}$ is the average rating of all the items of user u_2 (cf. [9, p. 738], [278, p. 153], [64, p. 619])"

Cf. [167, p. 67][181].

B.14. Challenges for Collaborative Filtering

New Item Problem

On most platforms, new items are frequently added to a *Recommender System* due to the publication of new products, seasonal offerings, and many other circumstances. "Therefore, until the new item is rated by a substantial number of users, the recommender system would not be able to recommend it" [9, p. 740]. In some cases, these items are presented to some randomly chosen users to collect as soon as possible the critically needed amount of information. Others use approaches from *Content-based Filtering* to compensate missing data: Schein et al. "make predictions for unrated items by using content data: averaging the set of content data (e.g. actors) that associate with an item (e.g. movie)" [238].

White, Grey and Black Sheep

The accuracy of the prediction of appropriate items depends first and foremost on the predictability of single users. Most people follow the mainstream and thus, show similar preferences – e.g., they may be interested in the latest Hollywood movies, especially in those that lead the common charts. Those users are called white sheep. Users, in contrast, who are regularly interested in niche items, such as low-budget productions, are called black sheep. They are using the long tail item catalog and, typically, represent a still relevant target group. *Recommender System*s can classify this special audience and recommend appropriate niche items as long as there is still a small number of similar users [194, p. 21]. However, the biggest challenge for *RSs* comes from some so-called gray sheep. A gray sheep represents a user that acts sometimes as white sheep and sometimes as black sheep, but in an unpredictable way. As they combine the interest in both, mainstream and niche items, these users show the highest failure rate of predictions. A good approach is to isolate this group from the others – since their feedback might also influence the prediction for the rest of the community – and treat them in a separate way: For instance by using only *Content-based Filtering*. As the accuracy of a system plays a key role:

In order to analyze a *Recommender Engine* in terms of performance, usability, accuracy and other important *Key Performance Indicator*s, Chapter 6 will introduce different concepts for an extensive evaluation.

Lonely Beach Recommender Problem

Recommender Systems work under different conditions that actively affects the recommendation score: for instance, spontaneous trends, seasonal re-occurrences of conditions or other influence factors. However, in some cases, the number of the same recommendation for different users has

[181] The paper extraction was written by Christopher Krauss and reflect the work of [9, 278, 64].

an impact on the appropriateness of recommendations. A good example is a travel *Recommender System* that predicts the bests sights close to the hotel. This system might also recommend beaches – and a critical attribute of a beach is represented by the number of visitors. Obviously, the fewer visitors – at the best no one –, the less noise, the less litter pollution and the better the beach. On the other hand, each recommendation of the same beach decreases its loveliness and thus, its recommendation score. This issue is present in many different application areas and must be tackled for each use case separately. Other use cases are, e.g., the prediction of alternative streets during a traffic jam in a navigation system, or entirely different in a video portal, the equal distribution of calls on different videos to avoid peaks on the same content that might overstress a *Content Delivery Network (CDN)*.

For the planned system, this issue can be neglected, as most learning objects, besides the mentioned videos, are based on unlimited resources and, thus, do not influence the recommendation quality.

Shilling Attacks

Although or just because *RS*s bring a substantial potential for e-commerce services, some of its users try to manipulate automatically generated recommendations. Competitors might rate some competing items very low – often with fake accounts – or their items very high. These attacks "affect the quality of the prediction for many users, resulting in decreasing overall user satisfaction with the system. Such threats may cost users' time and money and pose a serious challenge to the recommender system administrators" [66, p. 67]. Shilling attacks are hard to detect as the given user might be only very critical, and an exclusion of the concerned user has grave consequences. Chirita et al. have shown that malicious users do "exhibit special, noticeable rating patterns" [66, p. 73] regarding the number of prediction differences, the standard deviation in user's ratings, the degree of agreement with other users and the degree of similarity with top neighbors. While the detection of shilling attacks is a time-consuming task, it helps to increase the overall prediction accuracy and customer satisfaction.

Open Corpus Problem

On the one hand, the item catalog that is the base for most recommendations shows a vast but mostly finite number of elements, so for instance, for movie recommendations at Netflix or products at Amazon. All items within a service are called closed corpus. On the other hand, some services also provide recommendations of external sources. "The open corpus problem applies when an unlimited set of documents is given that cannot be manually structured and indexed with domain concepts and metadata from a community" [191] – so, possibly an almost infinite number of links in the *WWW* or other items that cannot be processed by a single *Recommender System*. Open corpus recommendations bring a lot of power to the prediction: e.g., the recommendation of up-to-date news or the latest music from various external databases – and for the service provider without being directly involved in the cost-intensive item aggregation process. However, the drawback comes from its unpredictability, regarding quality, data structure, and the performance

of the algorithms when processing unstructured data.

Performance and Scalability

The number of users, items as well as the choice of algorithms directly affect the computing power. CPU, as well as memory and other hardware resources, must be observed to avoid service breakdowns. Some algorithms might be distributable for a single calculation or at least for each new request. And several algorithms show different complexity levels that grow in various manners, e.g., logarithmic, linear or exponential. In item-based Filtering, for instance, the computational complexity "grows linearly with the number of customers that in typical commercial applications can grow to be several million[...]" [147, p. 247]. To avoid delays for the end-customer, some pre-processing steps can be performed offline. On the other hand, a user might be disposed to wait a couple of seconds as long as there is a reasonable explanation for the delay.

No Explanations Problem

Adaptive Systems aim at bringing more convenience to the offered service. In a theoretical world, users trust such a system and follow the predicted recommendations. In fact, many reasons make the community suspicious or just less convinced. Especially, since service providers follow their business agenda, users often fear the risks of being influenced – e.g., by editorial advertisements. There is a special research area targeting this Influence Analysis. It exists for specific use cases, such as visitor planning [236] [290], e-commerce [219] and Social Networks [45] – or even subdomains, such as psychological aspects in *Recommender Systems* [292].

From a customer point of view, most *RS*s operate as black boxes with some input data, e.g., the ratings, and some output data, the recommendations. Especially, when predictions fail and recommendations reference inappropriate items, users want to determine the reason for the failure and, at best, want to correct its cause. Also, "recommendation results might change automatically when the context of the user changes [... which] can be confusing to the user" [270, p. 331]. However, when services do not explain their recommendations, users get frustrated. Therefore, Herlocker et al. suggest an explanation facility for *Recommender Engine*s and identified four main benefits of this transparency [124, p. 243]: (1) justification of recommendations, (2) more user involvement in the calculation process, (3) education of the user in terms of recommender strengths and limitations as well as (4) better user acceptance. However, showing the whole sophisticated mathematical model to a user could overburden him. Thus, the way of presenting explanations needs to be evaluated for each application area independently. Herlocker determines "Rating histograms" as the most appropriate way to explain an average algorithm. "In addition, indications of past performance; comparisons to similar rated items; and domain specific content features, such the actors and actresses in a movie are also compelling ways to justify a high recommendation" [124, p. 249].

Usability

Nava Tintarev identified seven aims of *Recommender System*s and resulting guidelines to present recommendations comprehensively [264] – among others: transparency, effectiveness, persuasiveness, efficiency, and satisfaction. However, practical experiments seem to disagree with

some of these points. A total transparent service telling about the whole processed data, for instance, can scare the end customer (e.g., because of the vast amount of collected data) instead of boosting the understanding and so, it might cause adverse effects. Further evaluations need to be conducted in real life settings.

B.15. Sparsity Analysis

In 2012, the sparsity of 13 *Recommender Systems* of popular web services have been analyzed[182], see the Figure B.6. As can be seen, the amounts of users and items per service are usually unequal. Either there are more users than items. In this case, the item catalog might be experienced as limited – in fact, beyond a certain threshold, users are not capable of overlooking the whole repository. Nevertheless, they might not find appropriate products. Or, in turn, the service manages more items than users, so the probability that every item was rated by at least one user is relatively small. That corresponds to the so-called new item problem.

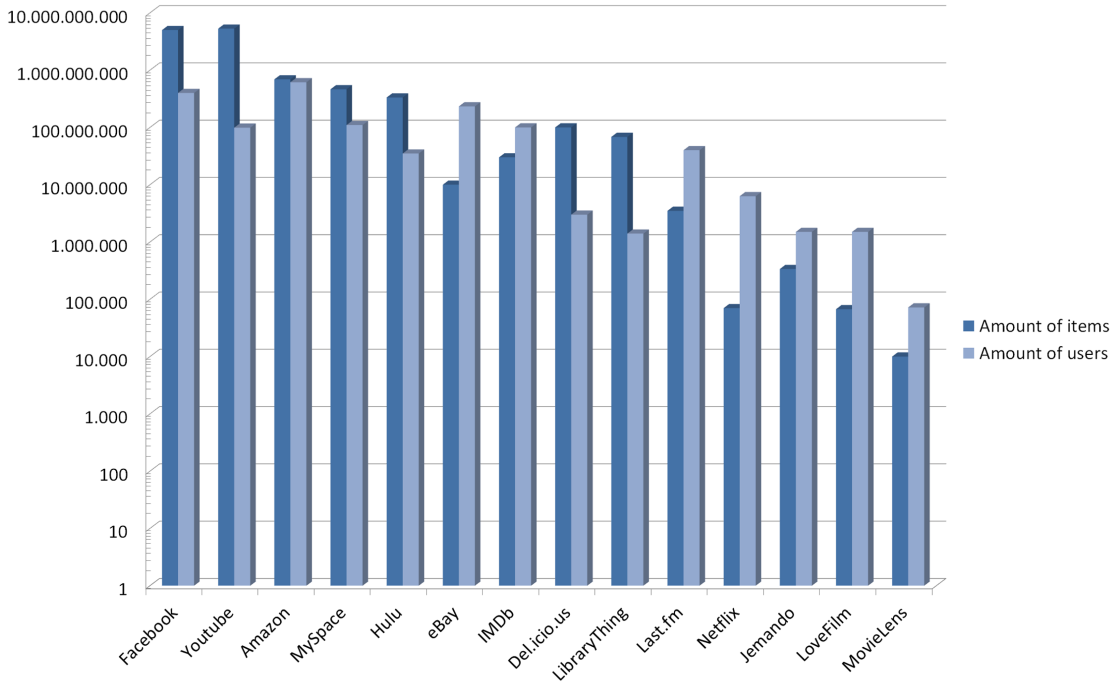

Figure B.6.: Sparsity Analysis from 2012 of popular services with Recommender Systems; y-axis: the number of items and users on logarithmic scales.

Figure B.7 indicates the possible size of the user-item-matrix (number of users multiplied by the number of items in light blue) and its actual density (in percent; dark blue) – so the real

[182] This analysis was part of the Master Thesis of Christopher Krauss with the title "Personalized Recommendations to be displayed on SmartTVs" (May 4, 2012).

number of user-item-associations within the service. The result: the sizes of the matrices are inversely proportional to their densities. That is due to the human ability to consume only a fixed amount of products. Just because persons have a broader choice, they are not going to consume more. With a certain amount of items, the number of feedbacks per user remains the same, independently from the catalog size. Consequently, with a constant community size, the more items, the fewer ratings per item.

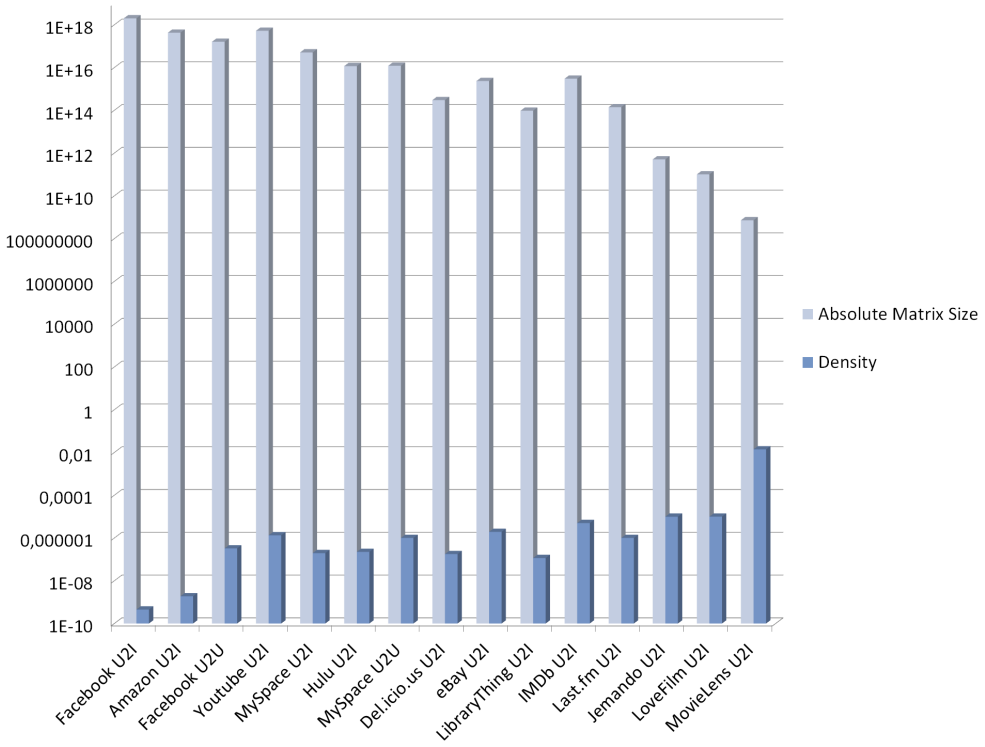

Figure B.7.: Sparsity Analysis from 2012 of popular services with Recommender Systems; y-axis: the feedback density (percentage of existing feedback compared to the total size of the user-item-matrix) on a logarithmic scale.

"Rashid et al. [222] classify different strategies to tackle the initial sparsity in user data by offering meaningful items to a new user during the registration process. The user will be asked to rate at least ten movies which were selected for specific strategic reasons, ranging from a random selection, to a popularity-based selection, through to a selection of the most valuable items for the recommendation algorithm. Unfortunately, this process still requires to browse between 50 and 80 different movies. As a consequence, over 13% of the users taking part in this experiment aborted the registration"

Cf. [163, p. 366] [183].

Another approach for compensating sparsity is to collect existing user data from other sources, such as social networks, where these users are already registered. Middleton et al. [196], for instance, used ontological inference in the user profiling process and so significantly improved the recommender performance by taking previously browsed research papers and their classifiers into account. The author of this dissertation conducted similar experiments, but on the TV program domain. The usage of a so-called Social Preference Ontology [163, p. 370] converts Facebook interests into *Recommender System* data. Thereby, likes but also written texts are analyzed to enrich the recommender's user profile. Especially Facebook comments are determined with the help of a novel Semantic Text Analyzer that identifies both: Semantic keywords representing the content as well as the sentiment, in terms of a numerical value that represents the author's opinion on the content. The extension of a Social Preference Ontology led to a prediction of interests in otherwise unknown items by 1.8% up to 15.1%.

B.16. Introduction of Different Learning Settings

Formal or Informal

Education is an essential task for humans as it allows to share and process masses of available information. Formal learning shall convey backgrounds and prepare for specific tasks, for instance known from classroom learning with teachers. Additionally, learners often take the control of their study conditions in *Self-Regulated Learning (SRL)* settings, where learners chose individually the topic, place and time of the knowledge acquisition. However, most people learn informally when trying to solve problems on their own because of personal or professional reasons [74]. Thereby, they are not necessarily aware that they are learning. *Life-Long Learning (LLL)* describes the pursuit of knowledge even after school. Obviously, there are a lot of different educational settings and their appropriateness depend on the topic, the teachers and especially the learners.

Online Learning

In addition to traditional *LMSs* for face-to-face courses and blended learning, Online-only Courses became very popular, primarily for higher education [258]. As universities hold their presence lectures anyway for their students, they are offering the same or similar contents over the *World Wide Web* for a broader target group. It is designed for blended learning courses with registered students of the same university [83], short-term students who just want to receive a specific certificate for the course participation or for guest students who are allowed to access the whole content, but without an official certification.

MOOCs

A *Massive Open Online Course (MOOC)* is a particular web-only course that is "open" for everyone and, as a result, is taken by a "massive" number of students. Three major *MOOC* platforms, founded in 2011 and 2012, share the market for higher education at the moment

[183] This extract was primarily written by Christopher Krauss. The main aim of the paper is to overcome the Cold Start Problem with the help of a Social Preference Ontology.

[258, 296]: Coursera[184] is driven by 149 partners, lead by people from Stanford University, and offers over 2,000 online courses for about 25 million learners. Udacity[185], born at Stanford University, as well, focuses on a lifelong learning experience and offers *MOOC* courses as well as industry-oriented "Nanodegree Programs". Until 2017, Udacity taught over 3m students and concentrates more on specialized contents and bridges the gap between education and required employment skills. Another popular *MOOC* service is edX[186], founded by MIT and Harvard, and offers over 1800 courses for over 10 million learners (over 33 million enrollments until end of 2017). Unfortunately, these educational organizations developed proprietary technical infrastructures, including self-developed *Learning Management Systems* containing a lot of sophisticated features, but as closed sources. Just edX is designed as an open, reusable platform called Open edX[187].

B.17. Ontology-based Educational Recommender Systems

In contrast to traditional *RS*, the classes of *Content-based Filtering*, as well as *Preference Filtering*, are under-represented because it would not make much sense to simply recommend similar learning objects or just learning objects that are preferred by the learner. Especially the word "preference" does not make much sense. Educational *RSs* are more about predicting learning objects that are of interest for reaching the course goals. That could be one reason, why the community tends to focus on semantic representations of knowledge in the user profile as well as semantic representations of learning contents and goals [207]. That could also be the reason, why most of the researchers introduce learning ontologies, like, for instance, Recker and Wiley did already in 2001. Their non-authoritative educational metadata ontology is designed to fit the needs of "filtering and recommending learning objects" [224]. Moreover:

> "The 'APOSDLE' Recommendation Service [251] uses an extended user profile as input for appropriate content recommendations and a web tool for ontology evaluation for identifying semantic similarities. The 'Multi-Attribute Recommendation Service' [193], in turn, uses ratings on different attributes and criteria for the same learning object in order to calculate proper recommendations"
>
> Cf. [161, p. 501][188].

Especially for open corpus recommendations, semantic tags are important and seem to bring a huge benefit for knowledge representations and further calculations, as shown in the area of "multilingual controlled vocabularies" across language and country borders [276]. And the analysis

[184] Coursera. See `https://www.coursera.org` (Accessed: 20.11.2017)
[185] Udacity. See `https://www.udacity.org` (Accessed: 20.11.2017)
[186] edX. See `https://www.edx.org` (Accessed: 20.11.2017)
[187] Open edX. See `https://open.edx.org/` (Accessed: 05.05.2017)
[188] This paper excerpt has primarily been written by Christopher Krauss.

of a "Simulation environment" [200] by Nadolski et al. result in more accurate – but maintain intensive – recommendations based on ontology-strategies compared to light-weight *CF* methods.

As already introduced in Section 4.2, even *xAPI* statements can be seen as an ontology on the learner's activity level, and *SCORM*, *LOM* and the like are ontologies on the learning content level. Thus, a *TEL Recommender System* that builds on these type of data can be classified as Ontology-based Learning Recommender, as well.

B.18. Practical Examples of Context-Aware Systems

At the moment, the most popular examples of context-aware systems are developments of the big US technology providers, like Apple, Google, Microsoft or Amazon: for instance, Google Now[189] is a digital everyday-life adviser. It can be asked to switch on the light in the living room or for the best route to work. Consequently, it processes mined contextual data, such as past and current geolocations. As a consequence, it knows the current position, predicts the assumed working place and the most appropriate route from one to another. Apple's Siri[190] or Amazon's Echo[191] work in similar ways but on other data sources. All of them are incorporating user information from their own services (geo-positions, user interaction data and so on), collaborative data from other people who use the same service (e.g., for predicting traffic jams) as well as external data from additional service providers (such as weather data).

B.19. Market Share of Learning Management Systems

LMS Market Share

Figure B.8 shows the evolution and market share of different *Learning Management Systems* from 2000 to 2016 in the European context. According to LISTedTECH.com, Moodle[192] is the most used system in Europe since 2006. In 2016, it had a market share of 65%. Blackboard Learn[193], as the main competitor, reached only 14%. Other services show a global market share of less than 5%. However, when focusing on different domains, such as higher education or industrial-driven training, or on specific markets, such as USA or Canada, there are different shares and the market seems to be more fragmented.

Figure B.9 shows the shift from *Learning Management Systems* to Learning Platforms for higher education in Northern America. It visualizes the foundation and, sometimes, merges of these platforms as well as a classification as open source technology. In the US and Canada, Blackboard has the biggest market share as primary *LMS* and merged with various platforms. However,

[189] Google Now. See `https://www.google.com/intl/de/landing/now/` (Accessed: 22.12.2016)
[190] Apple Siri. See `http://www.apple.com/de/ios/siri/` (Accessed: 22.12.2016)
[191] Amazon Echo. See `https://www.amazon.de/Amazon-SK705DI-Echo-Schwarz/dp/B01GAGVCUY` (Accessed: 22.12.2016)
[192] Moodle. See: `https://moodle.de/` (Accessed: 20.02.2017)
[193] Blackboard Learn. See: `http://www.blackboard.com/` (Accessed: 20.02.2017)

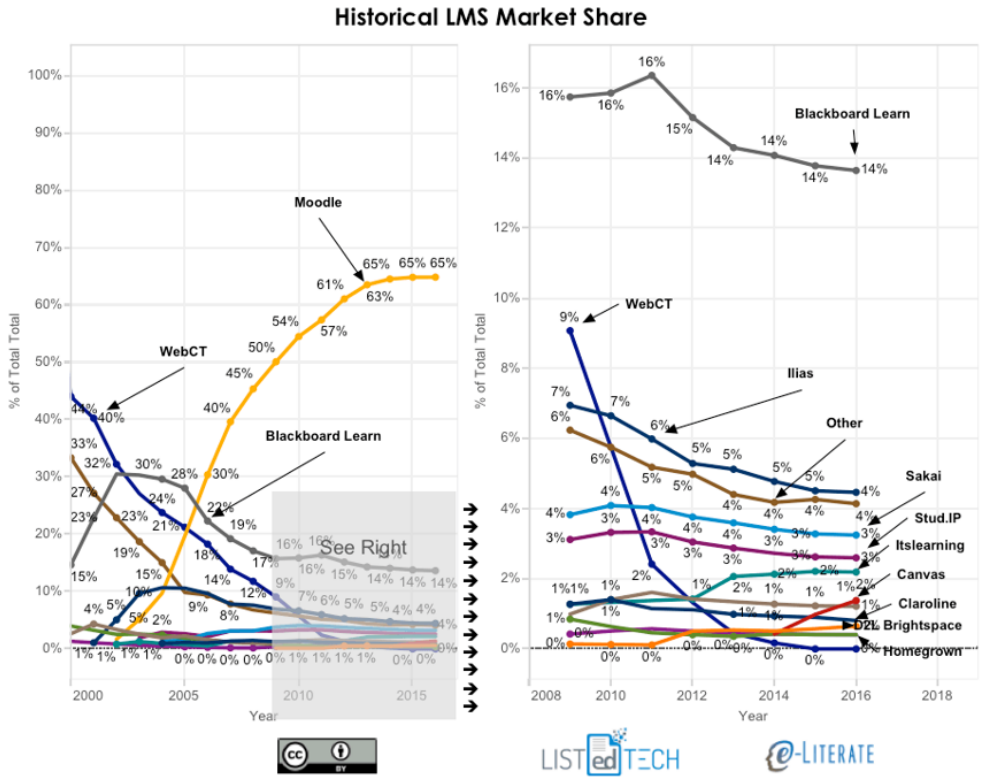

Figure B.8.: Historical Market Share of Learning Management Systems (2000 until 2016)
©LISTedTECH and e-LITERATE
see: http://listedtech.com/european-lms-market/ (Accessed: 13.02.2017)

Moodle is a prominent representative of the Open Source community as well as Canvas [194] that was just founded in 2010 and is growing massively. As one can see from Figures B.8 andB.9, the domain of *Learning Management Systems* has some market leaders, but changes regularly.

B.20. Additional Educational Specifications

Similar to *IMS* with its *Learning Resource Meta-data Specification*, another initiative started in 2011 standardizing descriptive data of learning material: The *Learning Resource Metadata Initiative (LRMI)* that builds on *Resource Description Framework (RDF)* [4]. *RDF* was created by the *W3C* and, today is an important component of the semantic web as it builds on concepts, such as subject, predicate, and object. With the help of *LRMI*, service providers can describe their contents with a huge vocabulary designed, first and foremost, for educational material. However, *LRMI* seems to be very flexible and light-weight[195] but not widely used, at least for higher education and by well-established *LMS*.

[194] Canvas. See: `https://www.canvaslms.com/` (Accessed: 20.02.2017)
[195] LRMI, Learning Resource Metadata on the Web. See: `https://blogs.pjjk.net/phil/lrmi-learning-resource-metadata-on-the-web-from-the-lile2015-workshop/` (Accessed 10.03.2017)

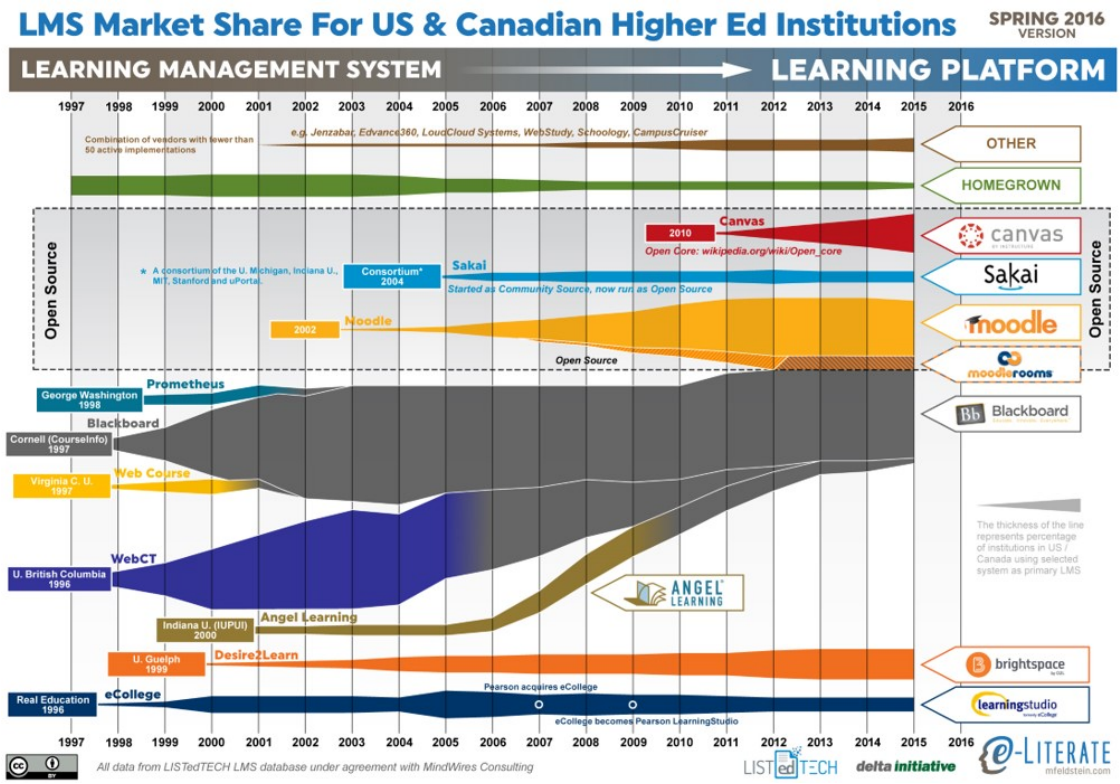

Figure B.9.: Historical Market Share of Learning Management Systems for US and Canadian Higher Education (1997 until 2016)
©LISTedTECH, delta initiative and e-LITERATE
see: http://mfeldstein.com/state-higher-ed-lms-market-spring-2016/ (Accessed: 13.02.2017)

An additional association, namely the *Aviation Industry Computer-based Training Committee (AICC)*, published specifications for computer-based training: *HTTP AICC Communication Protocol (HACP)* defines a way to integrate *HTML* learning contents into *LMSs*. However, this specification is less frequently used and, thus, can be neglected for this project.

B.21. Involved Stakeholders

This section introduces the set of important stakeholders and their roles roles in a typical learning environment. Roles can be taken over by single persons or multiple ones, depending on the learning context, the size of the institutional team and legal requirements. See Figure B.10 for an overview of the most important stakeholders with their core use cases in the smart learning environment. The architecture allows differing between 5 roles:

1. An administrator is the technical expert of the system, maintains the components and manages all user roles.

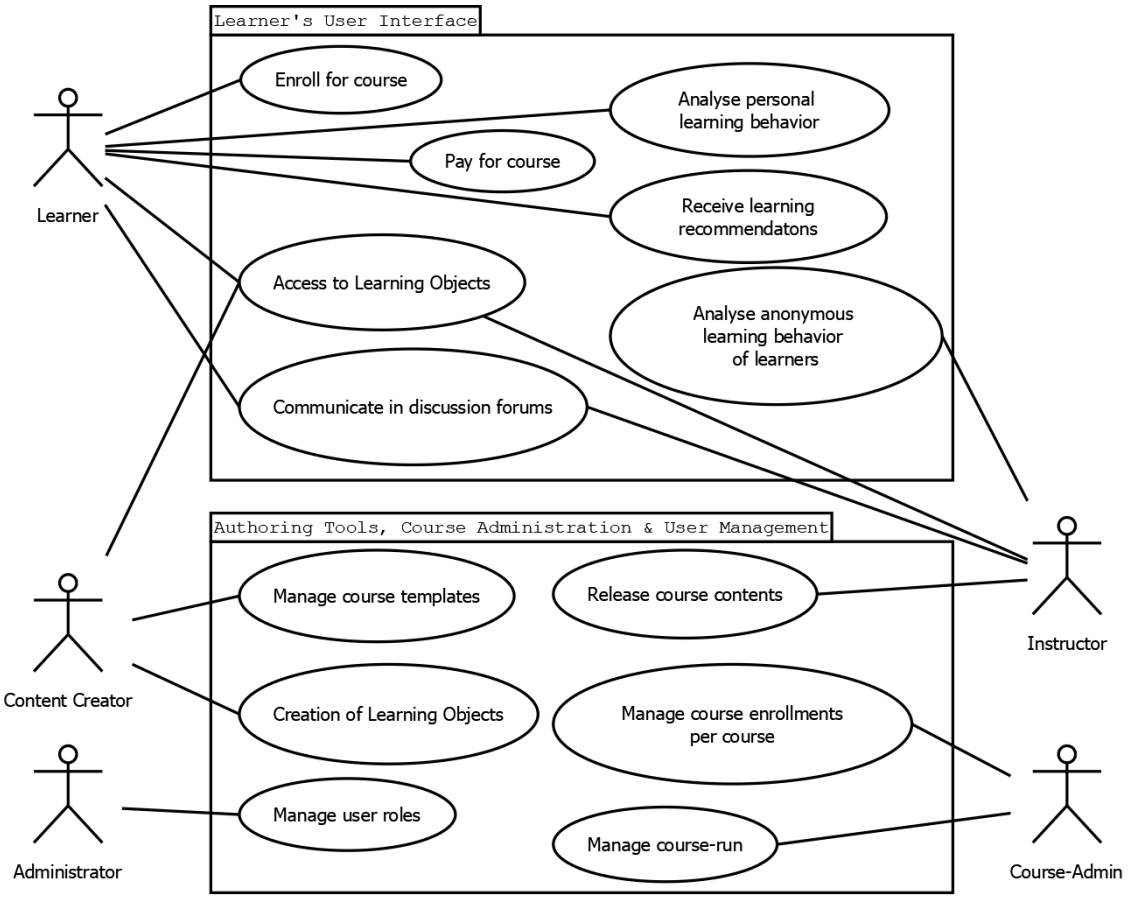

Figure B.10.: Use-Case-Diagram with the core actions that need to be performed by the different stakeholder of a digital learning environment

2. Content creators develop digital media that can be presented as *Learning Objects*. *Learning Objects*, in turn, can be bundled and structured to learning units and whole courses. For development reasons, they can also access the contents in the actual user front-end.
3. A course-administrator can create instances of a previously created course, the so-called course-runs. A course-run contains additional data, such as start and end date, a particular instructor and a list of enrolled students, which is also managed by the course-administrator.
4. An instructor is a direct contact with the learners. She or he might hold presence lectures and runs the actual course instance. Besides the communication with course participants, instructors can also analyze the learning progress and behavior of the learners.
5. Learners enroll in courses (e.g., after paying for it), access the *Learning Objects* and communicate via discussion forums. Moreover, with the help of a developed *Recommender System*, they can keep track of their learning behavior and receive personal recommendations to overcome learning weaknesses.

It is common sense that instructors can also take the role of content creators to develop new

and improve existing contents. Additionally they can overtake the role of course-administrators to manage course enrollments. However, if some institutions might limit the access levels of particular stakeholders (e.g., a guest lecturer should not get access to course enrollments), the actual roles are defined in a more restricted way.

B.22. FOKUS-Akademie Front-end

Besides the *LCA*, the learning infrastructure also allows other *LMSs* to access once created *Learning Objects*. One example is the "FOKUS-Akademie"[196], which builds on the Open Source project "Open edX"[197] and was adapted for the use at *FOKUS*.

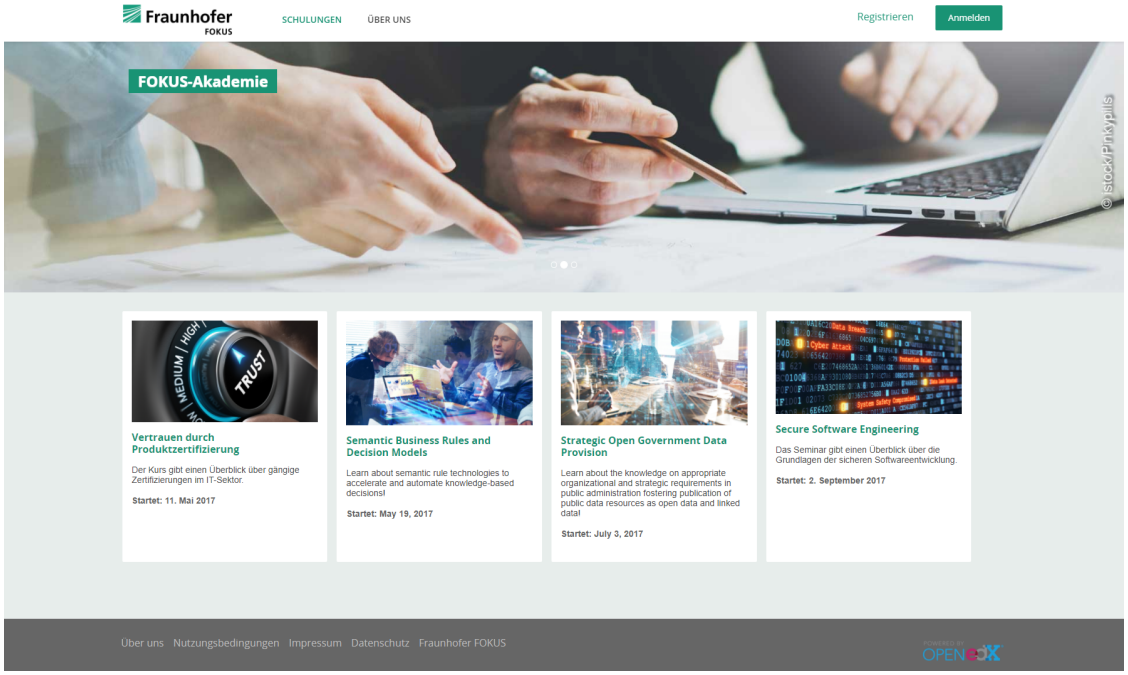

Figure B.11.: Overview of available courses at the FOKUS-Akademie on a desktop Chrome browser.

Optional Features

Open edX brings a fully-fledged *Graphical User Interface* that is used by millions of users in its original version edX (see Section B.16). It is available on Git and can be hosted by every interested institution. In addition to the features of the *Learning Companion Application*, it allows managing course enrollments for both educational staff and the learners directly within the platform. Its further advantages come from some available components that can be activated as optional features: among others, a sophisticated authoring tool (called Studio), Learning Analytics

[196] The FOKUS-Akademie (English-translation: Academy of *Fraunhofer-Institute for Open Communication Systems*) is a *LMS* designed and developed by a team of *FAME* led by Christopher Krauss. See `https://akademie.fokus.fraunhofer.de/` (Accessed: 06.07.2017)

[197] Open edX. See: `https://open.edx.org/` (Accessed: 06.07.2017)

and even Payment-Processors that allow interested users to register, pay and enroll for a course without requiring any action of educational employees.

Open and/ or Proprietary

Open edX uses proprietary content formats – e.g., for exchanging contents between multiple Open edX instances. This content format is called *Open Learning XML (OLX)* and presents an extension of the *Extensible Markup Language (XML)* to be used in an educational context [198]. Course elements, in contrast, are tagged as special components, so-called XBlocks, and can contain different hierarchical levels up to a whole course. "Like HTML tags, XBlocks can represent pieces as small as a paragraph of text, a video" [199] and so on. To the best of the author's knowledge and belief, *OLX* and XBlocks are not supported by other common *LMSs*.

LTI and Open edX

As an alternative to *Open Learning XML*, the *Learning Tools Interoperability Specification* is supported in both directions: to access contents hosted in Open edX via the *LTI* provider interface in other *LMSs* and to integrate externally hosted *LTI* contents in the Open edX front-end as *LTI* consumer.

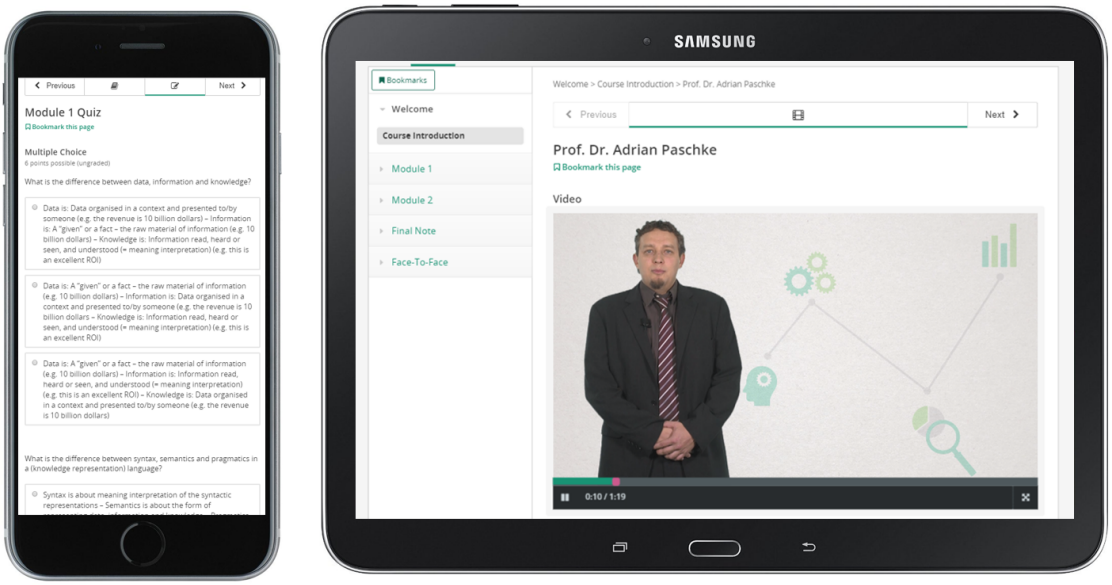

Figure B.12.: Content views (left: multiple choice quizzes; right: introduction video) for the course "Semantic Business Rules and Decision Models" at the FOKUS-Akademie in mobile-simulated environments.

Custom Themes

The most expensive task, when developing the FOKUS-Akademie, was its branding to appear in the same corporate design as the official website of *FOKUS* as this required a lot of testing and optimization in different environments. Figure B.11 shows the desktop web version. Thereby, the project team also focused on the mobile design of the website to display the contents on tablets, smartphones, but also still in a desktop browser. This responsive development is an important

[198] See OLX Description at Open edX Website: `http://edx-open-learning-xml.readthedocs.io/en/latest/what-is-olx.html` (Accessed: 06.12.2016)

[199] See XBlocks in Open edX: `https://open.edx.org/xblocks` (Accessed: 06.12.2016)

aspect of the paradigm of mobile and situated learning. Thus, it allows anytime and anywhere access to the contents. Figure B.12 shows *Learning Objects* in a mobile-simulated environment on smartphones and tablets.

B.23. Graphical User Interface of the Learning Companion Application

Main Navigation

In order to allow an understanding of the collected data, the user interface is briefly introduced. Learners access course materials using the *LCA*. However, course enrollments are managed by administrators in a separate system. After logging in, learners and teachers see an institute-specific welcome note. The main menu refers to the following additional pages (see the left menu in Figure B.13 for a visualization of the *LCA* navigation in a Desktop Browser):

- "Learning Recommendations": This section gives an overview of different Top-N item lists in different sections (one per approach) and related descriptions for the recommendations. Clicking on each item refers the user to the content page.
- "Course Overview": Within this section, the user gains access to the course contents in the enrolled courses. The contents are structured in different hierarchy levels and presented in a tile-based layout per level. See the content on the right side in Figure B.13 for all learning units within the section "Bestandsaufnahme".
- If a user with the role of an instructor (teacher) logged into the *LCA*, an additional navigation point appears that is not available to students: the "Teacher's View". This presents an easy-to-use configurator that allows for an analysis of the students' performances anonymously.
- The next page allows for particular items to be searched for independently of the hierarchy level.
- "Help" introduces the components and their meaning of the *Learning Companion Application* in textual form.
- "Imprint" is needed by German Law and can be individualized by the educational institution.
- Also an important aspect of German Law is the "Data Privacy Statement" which can also be customized.
- Finally, the user can logout. If the user does not use that button to exit the app, the user's session will be stored in a cookie to be reactivated for the next usage.

Tile Layout

The item levels courses, course sections and *Learning Units* are presented in a tile layout. An *LU* contains a group of low-level *Learning Objects*. The tiles are ordered in the didactic structure given by the educational staff (from left to right, top to bottom) and are only presented in sets that

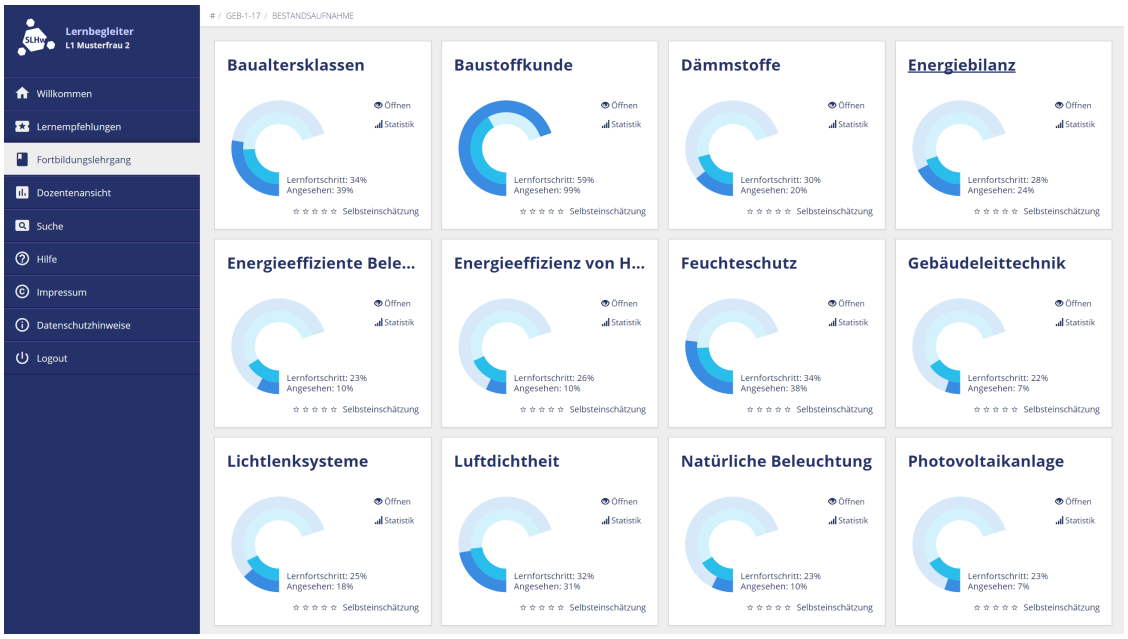

Figure B.13.: Screenshot of the tile overview of *Learning Units* (for the topic "Bestandsaufnahme" of the Energy Consultant Training) at the *Learning Companion Application* on a desktop browser.

belong to the same heading. Each tile contains a headline indicating the topic, a circle bar and some buttons. The circle progress bar shows two default values: The amount of consumed/learned elements in percent as well as a calculated learning need. The idea is to allow the learner to determine their learning progress at a glance and to quickly compare different aspects of the same heading. The more complete the bar, the better the learning progress and the lower the learning need. However, these two features are just examples and every content-related feature can be presented in the circle bar. Below the circles is a visualization of the learner's self-assessment on a one-to-five stars scale (zero means no self-assessment given, one stands for absolutely no knowledge and five stands for expert knowledge). The user can click on each star at any hierarchy level – the more, the better. However, if the user did not provide any self-assessment, it is implicitly transferred and averaged from lower-level items.

A click on the learning statistics opens a detailed view of all collected data related to the selected content. This corresponds to the learning dashboard approach proposed by Duval who visualized "the total time spent on the course, the average time spent on a document, the number of documents used and the average time of the day that the students work" [97] to the learner. The data of that page come from an external module, in which case from the connected *Recommender System*. A click on a content tile or on "open" initializes the next hierarchy level (including an animation). At the lowest level, an alternative view with a list of all *Learning Objects* contained by the selected *Learning Unit* is presented.

Figure B.14.: *Learning Companion Application* in mobile-simulated environments; left: presentation of a *Learning Object* of type text; right: statistics with personalized course progress and predicted knowledge levels (for the topic "Luftdichtheit" at the Energy Consultant Training).

Content Accordion

All *LOs* belonging to the same *Learning Unit* are displayed on one page in a flat list – one below the other and with a small icon indicating the media type of the offered *LO*. The left side of Figure B.14 visualizes the app on a smartphone with three *Learning Objects*. This set of items is integrated as an *LTI* tool[200]. A click on one *Learning Object* expands the content that is typically short enough for mobile learning. At the same time, the previous *LO* is closed. This effect of expanding one item and closing the others is reminiscent of an accordion – which is the reason why it is called the accordion view. Thus, the learner focuses only on one low-level item, and every user interaction refers to this one item. Consequently, the system triggers only *xAPI* statements which exclusively refer to the *Learning Object* that is currently learned.

Learning Statistics

Every *Learning Unit* offers individual learning statistics. Thereby, all the collected data of the current user for the related item are visualized and described to allow the users to analyze their learning progress, render the calculation of recommendations understandable and increase the trust in the system. See the right side of Figure B.14 in the tablet view for a visualization of the statistics.

B.24. Instance of an xAPI statement

```
1 {
```

[200] The middleware component for presenting *LTI*-compliant *Learning Objects* is developed by the partners of Beuth University.

```
    ''version'': ''1.0.0'',
    ''verb'': {
        ''id'': ''http://adlnet.gov/expapi/verbs/initialized''
    },
    ''actor'': {
        ''objectType'': ''Agent'',
        ''mbox'': ''mailto:christopher.krauss@fokus.fraunhofer.de''
    },
    ''object'': {
        ''objectType'': ''Activity'',
        ''id'': ''https://vfh143.beuth-hochschule.de/fokus/fame/openStudio/
            middleware/repository/modules/SW50cm9fQWJvdXQ='',
        ''definition'': {
            ''type'': ''http://adlnet.gov/expapi/activities/module'',
            ''name'': {
                ''de-DE'': ''\u00dcber+Fraunhofer''
            }
        }
    },
    ''context'': {
        ''platform'': ''slhw.fokus.fraunhofer.de'',
        ''contextActivities'': {
            ''parent'': [
                {
                    ''objectType'': ''Activity'',
                    ''id'': ''https://vfh143.beuth-hochschule.de/fokus/fame/
                        openStudio/middleware/repository/courses/QVdU'',
                    ''definition'': {
                        ''type'': ''http://adlnet.gov/expapi/activities/course''
                    }
                }
            ],
            ''grouping'': [
                {
                    ''objectType'': ''Activity'',
                    ''id'': ''https://vfh143.beuth-hochschule.de/fokus/fame/
                        openStudio/middleware/repository/courses/QVdU'',
                    ''definition'': {
                        ''type'': ''http://adlnet.gov/expapi/activities/course''
                    }
                },
                {
                    ''objectType'': ''Activity'',
                    ''id'': ''https://vfh143.beuth-hochschule.de/fokus/fame/
                        openStudio/middleware/repository/modules/SW50cm9fQWJvdXQ
                        ='',
```

```
                    ''definition'': {
                        ''type'': ''http://adlnet.gov/expapi/activities/module''
                    }
                }
            ]
        },
        ''extensions'': {
            ''https://slehwr.beuth-hochschule.de/xapi/extensions/keywords'': [
                ''openLCMS'',
                ''openPresentation'',
                ''Module''
            ],
           ''https://slehwr.beuth-hochschule.de/xapi/extensions/systeminfo'': [
                ''screen: 768 x 1024'',
                ''browser: Safari'',
                ''browserVersion: 602.1'',
                ''browserMajorVersion: 602'',
                ''mobile: true'',
                ''os: iOS'',
                ''osVersion: 10.3.2'',
                ''cookies: true''
            ],
            ''http://adlnet.gov/expapi/activities/course'': [
                ''2016-10#AWT''
            ]
        }
    },
    ''authority'': {
        ''objectType'': ''Agent'',
        ''name'': '''',
        ''mbox'': ''mailto:truong-sinh.an@beuth-hochschule.de''
    },
    ''stored'': ''2017-06-30T09:18:06.887700+02:00'',
    ''timestamp'': ''2017-06-30T09:18:06.887700+02:00'',
    ''id'': ''ed3e0ff5-587f-48db-8577-4aca56727989''
}
```

Listing B.1: "Instance of an xAPI statement where Christopher Krauss (Agent: christopher.krauss@fokus.fraunhofer.de) opened (Verb: initialized) a content 'about Fraunhofer' (Object: 'über Fraunhofer') in the 'AWT' course at TU Berlin"

B.25. Activity Data of the Energy-Consultant Training

The activity data of the first two course runs of the Energy-Consultant Training at the Chamber of Crafts is presented in Figure B.15 and B.16. The x-axis displays the days of the course and the

y-axis the number of activities (per day) due to triggered *xAPI* statements.

Figure B.15.: Learning Locker Activity Chart for GEB September - December 2016 (Number of xAPI Statements per day in blue)

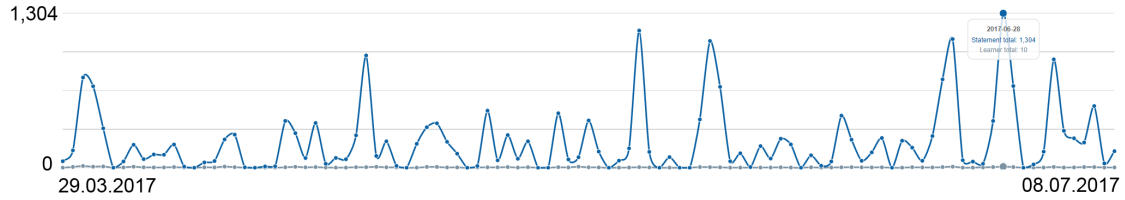

Figure B.16.: Learning Locker Activity Chart for GEB March - July 2017 (Number of xAPI Statements per day in blue)

B.26. Activity Data of the JavaFX Course

Figure B.17.: Learning Locker Activity Chart for JavaFX (Number of xAPI Statements per day in blue and number of active users per day in gray)

Similar to the data visualization of the Energy-Consultant Training, Figure B.17 presents the *xAPI* statements collected in the Java FX course.

B.27. Activity Data of the AWT Course

The reduced time interval in Figure B.18 (October 24, 2016, until January 22, 2017) indicates that there was even significant action in the *LCA* right before the exam in the course *Advanced Web Technologies*.

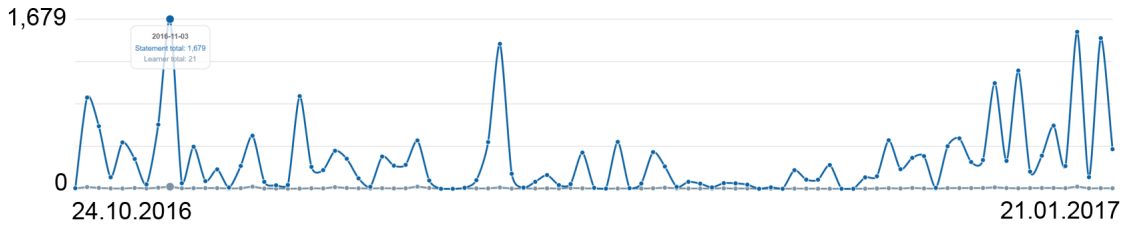

Figure B.18.: Learning Locker Activity Chart for AWT until three weeks prior the final exam (Number of xAPI Statements per day in blue)

B.28. Related Work on Typical Learning Patterns

Drop-Outs

With the establishment of *Massive Open Online Courses*, the barrier for a course enrolling by a potential participant is reduced to a minimum. At the same time, the huge set of participants led to high drop-out rates. Kloft et al., for instance, noticed a drop-out rate of 81.4% in a *MOOC* in the area of psychology with 11,607 enrollments at the beginning but only 3,861 participants at the end [156, p. 61]. At all, over 20,000 students engaged at some point in time within the 12 weeks course duration. They trained a *SVM* with different numerical features that describe users interactions at different points in time to forecast these drop-out rates – which worked well in the second half of the course.

Onah et al. even notice drop-out rates of up to 87 % (in some learning settings even more) in a survey of different publications [204, p. 5825 - 5826]. Thereby, the analysis of related work indicates that the course success rate decreases with higher participation numbers. They present the following reasons for drop-outs [204, p. 5828 - 5829]:

- Some students have no real intention to complete the course.
- They do not have enough time.
- The course is too difficult and lacks support.
- The student miss required digital skills, regarding media usage and self-learning-management.
- Some students had bad experiences with the content quality or with peers in the course.
- Students might have other expectations of the course.
- Some students might start too late and cannot catch up.
- Peer reviews as applied in some courses are experienced negative and show lower completion rates.

While dropout users represent a significant class of users due to their percentage and their negative impact, they represent just one type of learners. The rest of participants are users who engage with the items until they reach the course objective. Pardos et al. identified correlations between affect and behavioral engagement in a particular tutoring systems [208]. Therefore, they stated that the features frustration, confusion, concentration, boredom, gaming and off-tasks had effects on the final assessments.

Learning Patterns

> "Kizilcec et al. [155] investigated learners' engagement in courses from Coursera which consist of weekly videos and assessments, and proposed four typical engagement/disengagement patterns that they call
>
> - Completing: 'learners who completed the majority of the assessments offered in the class',
> - Auditing: 'learners who did assessments infrequently if at all and engaged instead by watching video lectures',
> - Disengaging: 'learners who did assessments at the beginning of the course but then have a marked decrease in engagement (their engagement patterns look like Completing at the beginning of the course, but then the student either disappears from the course entirely or sparsely watches video lectures)' and
> - Sampling: 'learners who watched video lectures for only one or two assessment periods'.
>
> These categories have been identified in three courses; however, their proportions differ in each course. To discover these categories, they have first characterized a student by a tuple giving her/his status each week: 'on track [T] (did the assessment on time), behind [B] (turned in the assessment late), auditing [A] (didn't do the assessment but engaged by watching a video or doing a quiz), or out [O] (didn't participate in the course in that week)' [155]"
>
> Cf. [17, p. 2][201].

The identification of the "completing" class is especially interesting for an educational *Recommender Systems*. Learners of this class show successful learning patterns for their learning methodology. While it does not necessarily represent the best or the only way of completing a course, it at least shows a higher probability of success. Thus, to reduce drop-outs, it might help to identify "completing" users and treat them as model students. "Auditing", "sampling" and "disengaging" students, in turn, might need additional impulses for learning and should be treated individually. Other researchers developed a more fine-granular classification model:

> "In an attempt to replicate the work of [155], Ferguson and Clow [109] suggest that the methodology used to uncover typical learning behaviors in a *MOOC*'s context does not necessarily generalize to another *MOOC* adopting different elements of pedagogy and learning design. Since the courses analyzed in [109] follow a social constructivist pedagogy, Ferguson and Clow adapt the methodology of [155]. They consider also participation in discussions

[201] The section about related work was written by primarily by Prof. Dr. Agathe Merceron with the help of Truong-Sinh An and Christopher Krauss.

> and end up with 10 values to characterize the weekly status of a student, instead of the four values T, B, A and O introduced in [155]. They have identified the following typical learning behaviors: Samplers ('Learners in this cluster visited, but only briefly', similar to sampling above), Strong Starters ('these learners completed the first assessment of the course, but then dropped out'), Returners ('these learners completed the assessment in the first week, returned to do so again in the second week, and then dropped out'), Mid-way Dropouts ('these learners completed three or four assessments, but dropped out about half-way through the course'), Nearly There ('these learners consistently completed assessments, but then dropped out just before the end of the course'), Late Completers ('this cluster includes learners who completed the final assessment, and submitted most of the other, but were either late or omitted some') and Keen Completers ('this cluster consists of learners who completed the course diligently, engaging actively throughout' similar to completing above). The two approaches in [155] and [109] share the same principle of selecting a priori features that are sensible to describe a student's individual engagement, and then use K-means clustering to discover typical learning behaviors"
>
> Cf. [17, p. 2][202].

While these clustering approaches are primarily applied on online and massive online courses, the patterns seem to be appropriate for a transfer to blended learning and digital assisted face-to-face courses.

B.29. Comparison of Activity Data in Two Different Course Settings

Figure B.19 compares the data of the first course-run of the Energy-Consultant Training at the Chamber of Crafts and the first *LCA* iteration of the course *Advanced Web Technologies* at the *TU Berlin*.

B.30. Clustering of LCA Users

> "[There] are multiple sensible ways to compare students in their learning behaviors. Because [the presented] time schedule is purely indicative for students and all the materials are available from the start of the course, we compared students on how they have interacted with the course independently of time. [...]

[202] The section about related work was written by primarily by Prof. Dr. Agathe Merceron with the help of Truong-Sinh An and Christopher Krauss.

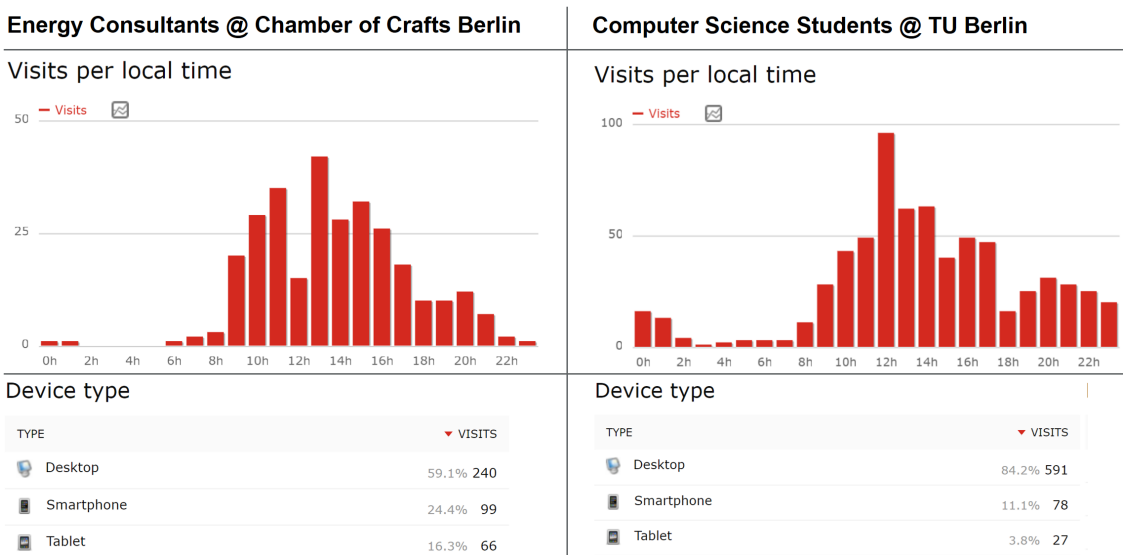

Figure B.19.: Comparison of the *LCA* usage between the Energy-Consultant (left) and the AWT participants (right). The data is extracted from PIWIK web analytics; top: activities over the daytime; bottom: hardware usage.

> We used RapidMiner[a] and applied the X-means clustering algorithm with Euclidean distance. X-means finds an optimal number of clusters and is known to find fewer clusters than K-means [214]. [...] To validate that the data does cluster naturally, we also applied K-means and checked for the drop in the curve plotting K against the sum of squared errors (which corresponds to the average within distance of RapidMiner)"
>
> Cf. [17, p. 3-4][203].

[a]Rapid Miner. See: https://rapidminer.com/

Four different promising settings have been applied for the clustering of the users. The clustering was applied after the end of the course with all interaction data collected during the course. Due to the random start center points of K-Means and the variable number of clusters in X-Means, we did not expect any predefined behavior classes – only a typical pattern that shows a high degree of similarity in the behaviors of comprised neighbors [204].

The four approaches of the learner clustering are:

- Clicks only: A student is represented as vector that comprises the number of clicks per *Learning Object*.
- Elapsed time: The student vector contains information on the total spend time per *Learning Object*.

[203]The section about methodology was written by primarily by Prof. Dr. Agathe Merceron with the help of Truong-Sinh An and Christopher Krauss.
[204]Details of the clustering approach are presented in Appendix B.30.

- Assessment scores: The student vector contains information on the normalized scores per *Learning Object*. *LO* scores a represented by ratings for self-assessments, answers in *QTI* tasks and JavaFX motivational feedback.
- Elapsed time and assessment scores: The student represented by a vector that comprises both, information on elapsed time and assessment scores (as described previously).

JavaFX Evaluation

In a second step after the determination of significant learning type clusters, the average marks per cluster are determined (only for the *AWT* course) to understand the degree of success per identified learning type better.

> "Because of the small data sets, particularly for the JavaFX course, clustering is challenging. We found that students do not act at random. In the JavaFX course, we could derive evidence behaviors that remind of patterns found in [155]]: completing, auditing, and disengaging"
>
> Cf. [17, p. 5][205].

AWT Evaluation

Particular interesting is the performance of the clustering approaches for the *AWT* course as the dataset also comprises numbers on the drop-out and exam participants as well as for the exam grades.

> "The first three approaches (Clicks only, Elapsed time and Elapsed time and assessment scores) lead to no meaningful results for the AWT course. [...] In contrast, for the Assessment scores approach, X-means generated three definite clusters. [...]
>
> Cluster 1 contains 9 students inclusive the one who did not pass the final exam [...]. Students in this cluster provided self-assessments in the first three units, and worked out exercises but did not achieve good scores. They remind of Strong Starters and Returners proposed in [109] when this vocabulary is adapted to the sequential order of the units instead of the first weeks of the course. To some extent, they exhibit also some kind of completing pattern in terms of exercises, because they completed almost half of them: on average 22 from a total of 48. Their average mark in the final exam is 2.03 which is slightly worse than the general average of 1.90.
>
> The biggest cluster contains 64 students [...] and is similar to the pattern auditing because they did exercises infrequently if at all: on average 1 out of 48. However, they did access .pdf files. All learners who did not participate in the final exam fall into this cluster. The average mark of the students in this cluster who participated in the final exam is 2.23 (no-shows are neglected), which is below the general average.
>
> The last cluster contains 26 students and shows a completing pattern [...]. If one sorts

[205] The discussion section was written by Truong-Sinh An, Prof. Dr. Agathe Merceron and Christopher Krauss.

the students according to the number of distinct exercises they have solved in the course, 25 of these 26 students are the top 25. They have worked on nearly all the exercises, on average 42 out of 49, and completed almost all of them correctly. The final exam mark in this completing cluster reaches 1.50 on average, a better mark than the overall average of 1.90"

Cf. [17, p. 5][206].

B.31. Related Work on Course Grade Predictions

Student Performance

Elbadrawy and Karypis showed that course results of already ended courses might also help to predict course grades of new starting courses. Those predictions, in turn, are valuable feedback for the course-selection process and can be incorporated in a course selection *Recommender System* [101]. Janssen et al. [139] proved the significant role of completion rates of learning objects in *TEL* recommenders.

Thai-Nghe et al. [262] note that a *Recommender System* based on linear regression (Matrix Factorization) can better predict student performances than traditional regression methods, such as logistic regression. Therefore, they utilized the KDD educational dataset [249] and improved the *Root Mean Square Error* by up to 3.7% [262, p. 2817].

Cortez and Silva showed that data mining techniques could be utilized to predict (secondary school) student performances [69]. They compared different methods that work well for specific courses. However, the accuracy is highly dependent on the kind of approach (they tested Decision Trees, Random Forest, Neural Networks and Support Vector Machines) as well as from the feature set selection in each course. A similar kind of research is carried out by Angel F. Agudo-Peregrina [12], who analyzed the interactions in their *LMS* for predicting students' academic performance. The prediction model is based on the relation feedback for student & student, student & teacher as well as student & content, the frequency of use (access to learning resources, the creation of class interactions) and the participation mode in the class (active or passive). They showed that student performances could also be approximated with the help of other sources, such as student and teacher activity in discussion forums [206].

B.32. Scientific Quality Criteria

Helmut Balzert formulated a catalog of quality criteria for science ethics [28, p. 13-15], which will be the basis for further research and experiments. The following list introduces Balzert's 12 central scientific quality criteria and their meaning for this dissertation:

[206] The analysis of the AWT results was written by Christopher Krauss, Prof. Dr. Agathe Merceron and Truong-Sinh An.

1. Relevance: The experiments shall create new insights and knowledge on a specific topic. Thus, only studies, which have not been published previously, are of interest to be conducted here. Relevant in the area of *TEL* is, what can be applied in real-world scenarios, such as courses with real participants.
2. Originality: This research aims at developing novel algorithms by following new ways. However, it is not the idea to invent something different compared to established research that has been introduced in the last Chapters, nor to just string together existing approaches. The dissertation identifies gaps and creatively transfers new and existing approaches to solve the task of appropriate learning recommendations.
3. Honesty: The research will be described truthfully and comprehensible what has been done.
4. Fairness: Fairness stands for a respectful treatment of work of colleagues, competitors and other researchers. External ideas will be credited as such.
5. Understandability: The research must be explained understandably.
6. Verifiability: Each approach shall be described in a way that it can be verified by external experts. Therefore, it will be elaborately described and published – as long as it does not create conflicts with the criteria of responsibility.
7. Responsibility: The author is responsible for her / his research, possible users and their privacy, the gained data and its security as well as for social, scientific and global impacts.
8. Reliability: All approaches must be reproducible in a way that other researchers gain the same results by reverse engineering the experiments.
9. Objectivity: Research must be impartial and neutral. For comparisons, the same evaluation metrics need to be used. Moreover, as student behavior will be analyzed, each student must be treated in the same way.
10. Validity: Validity stands for the degree of accuracy of an approach. In the data science community, different values have been established to validate algorithms, such as *Mean Absolute Error* and *Root Mean Square Error*. These values are described in this Chapter.
11. Logical reasoning: A conclusion should be based on significant assumptions and confirmed hypotheses. Therefore, the discussion Chapter will examine the hypotheses based on conducted experiments and draw appropriate conclusions.
12. Confirmability: Confirmability is the result of all appropriate followed quality criteria which have been listed above.

B.33. Hypotheses Require Quantitative or Qualitative Evaluations

To prove the correctness of the developed hypotheses, different evaluations are needed. This work uses deduction in an initial phase, where a theory and resulting hypotheses are evaluated in a top-down manner – thus, in the beginning, there is a formulation of a pattern that is derived from observations in multiple iterations. A theory that overcomes a set of falsification attempts is known as a particular valid theory. In turn, only one proofed falsification is enough to make an assumption invalid [28, 265, 44]. At the same time, deductive evaluation is used to observe inductive side effects. That are patterns which occur by accident but result in a new observation or a new theory [28].

The theory can be subdivided into a set of hypotheses. Some hypotheses require a qualitative evaluation of the object of investigation, regarding a deep analysis of some aspects [44, pp. 295]. For instance, the acceptance of the recommendations by users will be observed with think-aloud sessions, where learners perform predefined tasks at the system, while they have to comment each step. Both, their interaction on the screen, as well as their expressed comments, are being recorded. The qualitative methods help to observe complex phenomena, which cannot be reduced to a set of pre-given answers [28].

Quantitative evaluations, in turn, are needed to prove hypotheses in a deductive approach, first and foremost with numerical results – for instance, by comparing a considerable set of survey answers or by comparing statistical measurements of the algorithms [44, pp. 137].

B.34. User Studies

Even when proving the appropriateness of a system by performing algorithms measurements, subjective criteria are equally important. The human perception in general, but also of *Recommender Systems*, often is affected by external conditions – for instance by the *Graphical User Interface*.

Usability Effects

In the domain of TV program guides, a *Recommender System* was developed and applied at two industry partners utilizing two different user interfaces. Movisto[207] is an application that was designed to display exclusively top-7 program recommendations for different situations (now, upcoming tonight, highlights of the day and a personalized program guide). The multithek, in contrast, uses the same *Recommender System* for an Electronic Program Guide displaying all available programs per channel on a time axis. The user can hide inappropriate items by applying a visual filter. Figure B.20 shows the two services (left: Movisto and right: multithek) based rating

[207] Movisto: Service of RTV Arvato a company of Bertelsmann.

predictions for the same item metadata. Both services are based on user ratings, Movisto on a 1 to 10 rating scale and the multithek on a 1 to 5 stars scale. See [167] for a detailed description of the algorithms.

Figure B.20.: User Unterface of Movisto (left) and multithek (right) which are based on the same recommender algorithms that incorporates rating predictions.

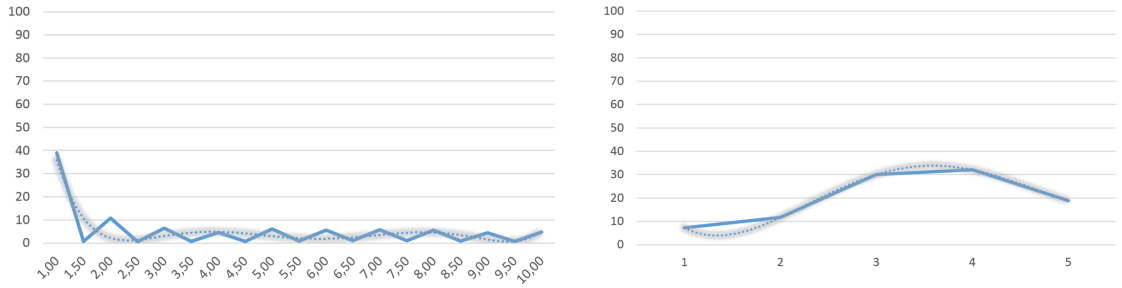

Figure B.21.: Rating scatter for: Movisto (left) and multithek (right). The x-axis shows the available ratings, and the y-axis indicates the amount per rating of all users (in percent) within the service.

Surprisingly, the gained rating data of both services diverge as shown in Figure B.21. While the multithek shows a rating scatter that reminds on a Gaussian distribution, the Movisto service shows an almost equal distribution of all ratings except for the lowest possible rating – the one with a very high percentage. Over one-third of all ratings refer to the lowest possible. After some considerations, an effect of the *Graphical User Interface* was identified that represents the most important influence factor: The list of top-7 recommendations seemed to be too limited to present the whole set available programs per situation. That is why users tend to give the lowest possible rating for items they do not like to receive a new list of top-7 recommendations. However, the not-liked items are not necessarily seen as totally inappropriate, but to make sure they do not appear again, they are rated low. Similar to these findings, researchers often noted the critical effect of user interfaces on the actual perception of *Recommender Systems* [255, 240, 264, 270].

Importance of User Studies

Usability is only one example of the possible effect of external conditions that directly affect the users' perception of a *Recommender System* which cannot be determined with algorithm measurements. Therefore, qualitative user studies as known from human-computer interaction research need to be conducted. However, these studies have been rarely applied for *Recommender*

Systems in *TEL* [191, p. 405].

In the general recommender domain, some research analyzed subjective factors that might influence the users' interaction with the system. Cosley et al. [70], for instance, observed discrepancies when users re-rated previously rated items again (1) on the same rating scale (mean deviation of 1%) and (2) on new rating scales (mean deviation of up to 12%). Finally, they also presented average rating values (mean deviation of 1%) and manipulated average rating values of ±1star for the re-rating process. The latter shows a mean rating deviation of up to 30% (+16% for and -14%) which indicates the ease of psychological interference by the user interface – an argument to perform user studies.

For a *TEL Recommender System* also the overall students' satisfaction plays a key role [68], [18] for the acceptance of a service. Kirkpatrick developed a model in the 1970's to evaluate training programs [154, p. 486] that consists of four layers of subjective user acceptance which can also be used for a qualitative evaluation of *Recommender Systems* as suggested by Manouselis et al. [191, p. 407]

1. Reaction: What thoughts and feelings occurred using the adaptive system, here including knowledge level predictions and learning need approximations as well as learning recommendations?
2. Learning: How helpful was the adaptive system in solving learning tasks?
3. Behavior: Does the adaptive system influence positively or negatively the behavior within and outside of the system?
4. Results: Does adaptive system improve the intended results, concerning reaching the learning objectives?

Acceptance Criteria

For instance, different factors of students' acceptance of Moodle have been analyzed [76], [39], [153]. Thereby, a study indicates that students show a higher acceptance "regarding ease of use, usefulness, access, reliability, and compatibility of the LMS than the educators" [137].

Types of User Studies

Some approaches have often be applied to evaluate *Recommender Systems* in a qualitative manner (cf. [125, 191, 270]).

- Questionnaires and Surveys can be conducted to get the general position on some aspects. The advantage lies in its scalability, as huge amounts of users can be asked (even at the same time when performing it online).
- Interviews, in contrast, help to get a better understanding of the individual opinion of a user, as the questioner can ask further questions on a particular topic if needed.
- In Thinking Aloud Studies, users are asked to perform a task with the help of the object of investigation. Thereby, the user should comment the perception and thoughts loudly while performing the task. As a result, usability and acceptance effects can be determined.

Recording the experiment (e.g., with Screencasts of the interaction with the software as well as camera-recordings for the comments of the users) is advisable for later confirmability.

- A/B Testing is the comparison of two settings of the same system, only the object of investigation is different per setting. For instance, El-Bisthouty et al. [100] compared the effect of their *Recommender System* for learning material on the user's knowledge-acquisition. Thereby, some tasks are performed in various phases with and some without the RS and the results, concerning performance in questionnaires, are compared to analyze the (here positive) effects of the approach.
- Trial experiments with analysis of Log Data, in contrast, can also give quantitative results, regarding general marks, drop-out rates, etc.

In this work surveys with course participants (prior, during and after the course), Thinking-aloud sessions for overall user acceptance when working with the system, comparisons of different *Learning Management Systems* and trial experiments are going to be performed. Moreover, more generalized surveys will be conducted with externals to analyze a broader spectrum of possible recipients of learning recommendations.

B.35. Example of an Increasing Time-Window Cross-Validation

Figure B.22 visualizes an example of learning activities with is utilized to explain the approach. While $User1$ only consumed one item ($i1$) at the very beginning and at the very end of the course, $User2$ studied $i2$ in regular intervals. Finally, $User3$ accesses a variety of items $i1, ..., i7$.

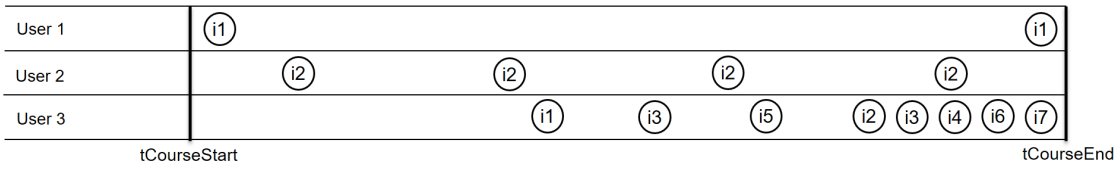

Figure B.22.: Example of activity data over a course period. $i1$ to $i7$ represent different items that have been consumed at particular points in time (x-axis) by the three users $User1$, $User2$ and $User3$.

For an increasing time-window cross-validation, the data can be split according to the threshold tr per evaluation step into training sets Tr and test sets Te as shown in Figure B.23.

At the point in time of each threshold tr, a user-specific Top-N list can be recommended based on the data of the training set Tr. The recommendations can then be evaluated with the help of the test set Te.

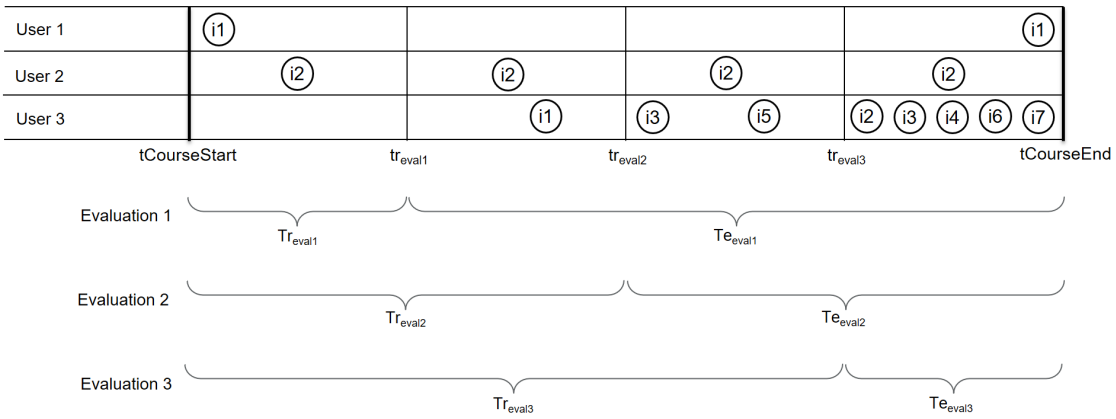

Figure B.23.: Example of data splits in an increasing time-window cross-validation with three item splits.

B.36. Additional Measures for Recommender Systems

Recently, additional measures have been introduced which are not focusing on accuracy or ranking precision tasks: among others novelty, diversity, sensitivity, and specificity as well as serendipity. However, these measures are listed for the sake of completeness but are optional for further evaluations in this research.

Novelty & Serendipity

New added items are of high interest for the users as they might represent the most up-to-date topics in an item catalog [125, 67, 133]. Unfortunately, it is tough to give these items a high probability of being recommended, because of the missing user feedback (see the New Item Problem in Section B.14). Hurley and Zhang presented one approach to analyzing the novelty of recommended items [133, p. 14:6]. In contrast to the proposed timeliness measure, a novelty value refers to the age of an item in the item catalog and not to the time span between the recommendation of an item and its usage. Since in our setting learning objects are rarely added to a course after the course start, this measure will be neglected. In addition to novelty, the idea of serendipity is to help "the user find a surprisingly interesting item" [125, p. 43] by adding the chance of presenting random items.

Sensitivity, Specificity & Coverage

Sensitivity indicates the "probability of a randomly selected good item actually being rated as good" and, thus, is similar to the meaning of precision. Specificity, in contrast, shows the "probability of a randomly selected bad item actually being rated as bad" [274, p. 12] which is almost the opposite. Coverage, in turn, corresponds to the share of calculated predictions out of all available items [274, p. 12]. However, as a course has a fixed amount of items, the calculation of predictions for all items would be executed in a reasonable time, and the coverage would stay constant.

Diversity

Hurley and Zhang [133] also noted that even a very accurate *RS* could dissatisfy the user, when its recommendations (e.g., in a Top-N item list) are very similar and refer to almost the same

contents. That is why the diversity measure indicates the variance in the recommended items [67]. The more diverse the items, the more satisfying the list of recommendations. However, this often comes with the effect of decreasing the overall accuracy [133, p. 14:2] and plays a more critical role in entertainment *Recommender Systems* with a high probability of presenting similar items than for *Technology Enhanced Learning*, where all *LOs* should be elementary distinguishable.

B.37. Example of Cleaned MATD

Figure B.24 presents an example of $MATD$, $tFirstConsumption$ and the resulting cleaned $MATD$. Only one data set splitting is presented for the sake of simplicity. The cleaned $MATD$ for $User3$ is 0, because $i3$ is the recommended item and at the same time the next consumed item.

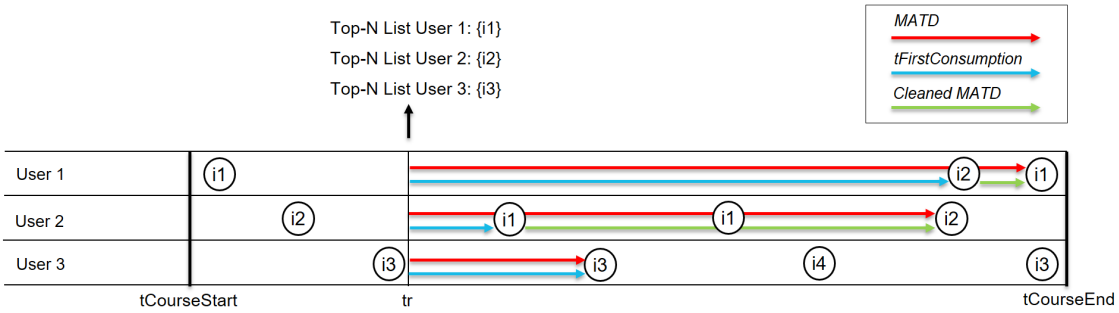

Figure B.24.: Example of the cleaned $MATD$ for the three users $User1$, $User2$ and $User3$.

B.38. Mathematical Derivation of the Cleaned MATD

tr is a equal for all K relevant items of the Top-N list – as well as tc_i. That is why both values can be either a part or not a part of the summation $\sum_{i=1}^{K}$:

$$
\begin{aligned}
MATD_{cleaned} &= \frac{\sum_{i=1}^{K} tc_i - tr}{K} - (tc_f - tr) \\
&= \frac{(\sum_{i=1}^{K} tc_i) - (tr * K)}{K} - (tc_f - tr) \\
&= \frac{\sum_{i=1}^{K} tc_i}{K} - \frac{tr * K}{K} - (tc_f - tr) \\
&= \frac{\sum_{i=1}^{K} tc_i}{K} - tr - (tc_f - tr) \\
&= \frac{\sum_{i=1}^{K} tc_i}{K} - tr - tc_f + tr \qquad \text{(B.5)} \\
&= \frac{\sum_{i=1}^{K} tc_i}{K} - tc_f \\
&= \frac{\sum_{i=1}^{K} tc_i}{K} - \frac{tc_f * K}{K} \\
&= \frac{(\sum_{i=1}^{K} tc_i) - (tc_f * K)}{K} \\
&= \frac{\sum_{i=1}^{K} tc_i - tc_f}{K}
\end{aligned}
$$

B.39. Evaluation Setting 1: AWT with LOs and LUs

According to the evaluation setting proposed by Campos et al. [57], the methodological questions are answered as follows for Evaluation Setting 1:

MQ1 *What base set is used to perform the training-test splitting?*
The 17 weeks *AWT* course data is used as base data set. The course ran from October 24, 2016, until February 12, 2017. The user-item matrix comprises the number of activities per *Learning Object* and per *Learning Unit* per user, e.g. a user who accessed 3 times the same item gets a score of 3 in the according user-item-tuple.

MQ2 *What [scoring] order is used to assign [relevance scores] to the training and test sets?*
The scoring is based on the algorithm's relevance score, which is introduced for each approach separately. The higher the predicted score, the more probable the item appears in the Top-N list. Top-N items are not ordered internally.

MQ3 *How [much data] comprise the training and test sets?*
The test data comprises all 99 active users and 44,421 *xAPI* statements on all 1,006 *Learning Objects* (including the automatically generated one, such as for self-assessments) and on the 106 *Learning Units*. Table B.1 presents the composition of the training and test data for the different week splits.

MQ4 *What cross-validation method is used for increasing the generalization of the evaluation*

results?

The course is split according to the "increasing time-window" validation into 17 sub-datasets where the time threshold shifts by seven days and is defined per week to be on Mondays at 0 pm. Thereby, with each split, the duration of the training dataset increases by seven days, and the test dataset decreases by the same amount. Due to the cold start problem, no evaluation has been performed in the first week.

MQ5 *Which items are considered as target items (in a top-N recommendation task)?*

Items are determined according to the algorithm approach which is described separately. The value N defines the number of recommendations in the Top-N item list. The Number of N is given per approach.

MQ6 *Which items are considered relevant for each user (in a top-N recommendation task)?*

Items are relevant that have been processed by the same user after its recommendations.

Table B.1.: Training and test data splitting in Evaluation Setting 1

Splitting at the	Training Dataset		Test Dataset	
beginning of Week	Users	Statements	Users	Statements
1	0	0	99	44421
2	24	1210	99	43164
3	50	2657	98	41717
4	63	3874	96	40500
5	66	5556	95	38818
6	72	7128	92	37246
7	78	9281	91	35093
8	80	10345	90	34029
9	81	11935	88	32439
10	82	13132	88	31242
11	82	15178	88	29196
12	82	16542	88	27832
13	87	18907	87	25467
14	93	20627	86	23747
15	95	23344	83	21030
16	99	27299	75	17075
17	99	33786	63	10588

B.40. Evaluation Setting 2: AWT with LUs

According to the evaluation setting proposed by Campos et al. [57], the methodological questions are answered as follows for Evaluation Setting 2:

MQ1 *What base set is used to perform the training-test splitting?*
The 17 weeks *AWT* course data is used as base data set. The course ran from October 24, 2016, until February 12, 2017. The user-item matrix comprises the number of activities per *Learning Unit* per user, e.g. a user who accessed 3 times the same *LU* gets a score of 3 in the according user-item-tuple. The activities on the low-level *LOs* are not considered.

MQ2 *What [scoring] order is used to assign [relevance scores] to the training and test sets?*
The scoring is based on the algorithm's relevance score, which is introduced for each approach separately. The higher the predicted score, the more probable the item appears in the Top-N list. Top-N items are not ordered internally.

MQ3 *How [much data] comprise the training and test sets?*
The test data comprises all 99 active users and 8,241 *xAPI* statements on all 106 *Learning Units*. Table B.2 presents the composition of the training and test data for the different week splits.

MQ4 *What cross-validation method is used for increasing the generalization of the evaluation results?*
The course is split according to the "increasing time-window" validation into 17 sub-datasets where the time threshold shifts by seven days and is defined per week to be on Mondays at 0 pm. Thereby, with each split, the duration of the training dataset increases by seven days, and the test dataset decreases by the same amount. Due to the cold start problem, no evaluation has been performed in the first week.

MQ5 *Which items are considered as target items (in a top-N recommendation task)?*
Learning Units are determined according to the algorithm approach which is described separately. The value N defines the number of recommendations in the Top-N item list. The Number of N is given per approach.

MQ6 *Which items are considered relevant for each user (in a top-N recommendation task)?*
Items are relevant that have been processed by the same user after its recommendations.

Table B.2.: Training and test data splitting in Evaluation Setting 2

Splitting at the beginning of Week	Training Dataset Users	Statements	Test Dataset Users	Statements
1	0	0	99	8241
2	24	127	99	8114
3	50	369	98	7875
4	63	538	96	7708
5	66	729	95	7515
6	72	942	92	7301
7	78	1139	91	7107
8	80	1188	90	7058
9	81	1303	88	6940
10	82	1337	88	6906
11	82	1366	88	6876
12	82	1369	88	6873
13	87	1609	87	6636
14	93	1870	86	6377
15	95	2281	83	5962
16	99	3298	75	4945
17	99	5029	63	3215

B.41. Evaluation Setting 3: JavaFX

According to the evaluation setting proposed by Campos et al. [57], the methodological questions are answered as follows for Evaluation Setting 3:

MQ1 *What base set is used to perform the training-test splitting?*

The 12 weeks JavaFX course data is used as base data set. The course ran from December 4, 2016, until February 19, 2017. The user-item matrix comprises the number of activities per *Learning Object* and per *Learning Unit* per user, e.g. a user who accessed 3 times the same item gets a score of 3 in the according user-item-tuple.

MQ2 *What [scoring] order is used to assign [relevance scores] to the training and test sets?*

The scoring is based on the algorithm's relevance score, which is introduced for each approach separately. The higher the predicted score, the more probable the item appears in the Top-N list. Top-N items are not ordered internally.

MQ3 *How [much data] comprise the training and test sets?*

The test data comprises all 28 active users and 2,577 *xAPI* statements (3,624 in total and 2,566 in the 12 week course duration) on 4 *Learning Units* and 89 *Learning Objects* (including

the automatically generated one, such as for self-assessments). Table B.3 presents the composition of the training and test data for the different week splits.

MQ4 *What cross-validation method is used for increasing the generalization of the evaluation results?*

The course is split according to the "increasing time-window" validation into 12 sub-datasets where the time threshold shifts by seven days and is defined per week to be on Mondays at 0 pm. Thereby, with each split, the duration of the training dataset increases by seven days, and the test dataset decreases by the same amount. Due to the cold start problem, no evaluation has been performed in the first week.

MQ5 *Which items are considered as target items (in a top-N recommendation task)?*

Learning Units are determined according to the algorithm approach which is described separately. The value N defines the number of recommendations in the Top-N item list. The Number of N is given per approach.

MQ6 *Which items are considered relevant for each user (in a top-N recommendation task)?*

Learning Units are determined which have been accessed by the users after the point in time of the recommendation (data of the test set).

Table B.3.: Training and test data splitting in Evaluation Setting 3

Splitting at the	Training Dataset		Test Dataset	
beginning of Week	Users	Statements	Users	Statements
1	0	0	28	2577
2	15	340	24	2237
3	16	644	21	1933
4	17	949	18	1628
5	19	1187	15	1390
6	20	1426	13	1151
7	25	1933	7	644
8	26	2094	5	483
9	27	2255	4	322
10	28	2416	2	161
11	28	2566	1	11

B.42. Evaluation Setting 4: Energy-Consultant Training

According to the evaluation setting proposed by Campos et al. [57], the methodological questions are answered as follows for Evaluation Setting 4:

MQ1 *What base set is used to perform the training-test splitting?*

The 16 weeks Energy-Consultant Training data is used as base data set. The course ran from March 29, 2017, until July 8, 2017. The user-item matrix comprises the number of activities per *Learning Object* and per *Learning Unit* per user, e.g. a user who accessed 3 times the same item gets a score of 3 in the according user-item-tuple.

MQ2 *What [scoring] order is used to assign [relevance scores] to the training and test sets?*

The scoring is based on the algorithm's relevance score, which is introduced for each approach separately. The higher the predicted score, the more probable the item appears in the Top-N list. Top-N items are not ordered internally.

MQ3 *How [much data] comprise the training and test sets?*

The test data comprises all 15 active users (8 learners and 7 staff members) and 15,545 *xAPI* statements on 39 *Learning Units* and 366 *Learning Objects* (including the automatically generated one, such as for self-assessments). Table B.4 presents the composition of the training and test data for the different week splits.

MQ4 *What cross-validation method is used for increasing the generalization of the evaluation results?*

The course is split according to the "increasing time-window" validation into 16 sub-datasets where the time threshold shifts by seven days and is defined per week to be on Mondays at 0 pm. Thereby, with each split, the duration of the training dataset increases by seven days, and the test dataset decreases by the same amount. Due to the cold start problem, no evaluation has been performed in the first week.

MQ5 *Which items are considered as target items (in a top-N recommendation task)?*

Learning Units are determined according to the algorithm approach which is described separately. The value N defines the number of recommendations in the Top-N item list. The Number of N is given per approach.

MQ6 *Which items are considered relevant for each user (in a top-N recommendation task)?*

Learning Units are determined which have been accessed by the users after the point in time of the recommendation (data of the test set).

Table B.4.: Training and test data splitting in Evaluation Setting 4

Splitting at the beginning of Week	Training Dataset Users	Statements	Test Dataset Users	Statements
1	0	0	15	15545
2	11	1817	15	13728
3	11	2192	15	13353
4	11	2621	15	12924
5	12	3489	15	12056
6	12	4979	15	10566
7	13	5724	15	9821
8	13	6361	15	9184
9	13	6905	14	8640
10	13	7964	14	7581
11	14	9328	14	6217
12	14	10253	13	5292
13	15	11056	13	4489
14	15	12767	13	2778
15	15	14741	11	804

B.43. Slope One Extension – Effect of Wrong Implementation

In a first attempt to evaluate the effect of the equation in Section 7.1.3, the formula was accidentally adjusted as follows:

$$relevanceScore_{u,i} = realAccesses_{u,i} - predictedAccesses_{u,i} \tag{B.6}$$

Even though this $realAccesses_{u,i}$ were zero, it would change the relevance score to the opposite. The meaning behind this would be that the greater the gap is between the prediction and the personal number of consumption, the lower is the probability that this item is recommended. The effect is visualized in Figure B.25 and is not only observable by a low precision but also by a high timeliness value.

B.44. Time-Weighted Slope One evaluated on Learning Objects

The Time-Weighted Slope One algorithm is evaluated according to Evaluation Setting 1 based on *Learning Objects* (see Appendix B.39). The results of the first setting are presented in Figure B.26. As one can see, the time-based approach performs almost similar to the conventional algorithm for a decay power a of 1.

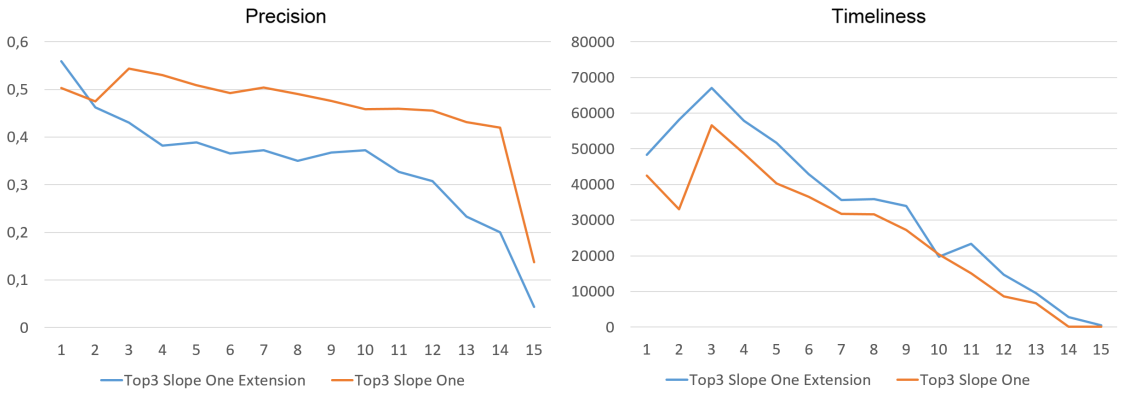

Figure B.25.: Accuracy of the extended Slope One algorithm in a false implementation (blue) and the traditional Slope One (orange) – left: average precision per week of the AWT course; right: timeliness values (according to $MATD_{cleaned}$) given in minutes

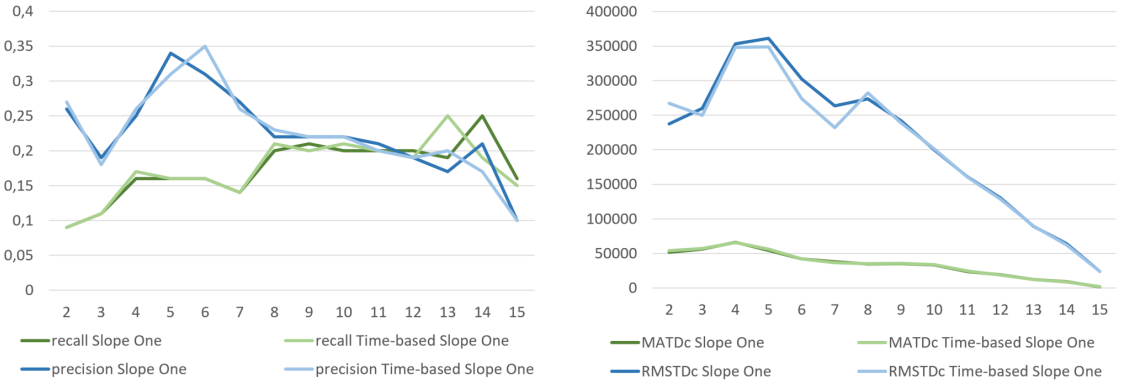

Figure B.26.: Slope One and Time-based Slope One accuracy comparison – left: average precision and recall; right: Timeliness values for averaged $MATD_{cleaned}$ and $RMSTD_{cleaned}$ given in minutes

B.45. Performance of the Standard and the Time-Weighted Slope One

Figure B.27 indicates the similar performance (in terms of computing time) for the standard and the time-weighted Slope One. However, weights with $\alpha = 0$ have been computed for this analysis which means the performance of the standard Slope One algorithm is expected to be faster when neglecting this computation step.

B.46. Number and Percentage of Repeated Item Accesses per Week

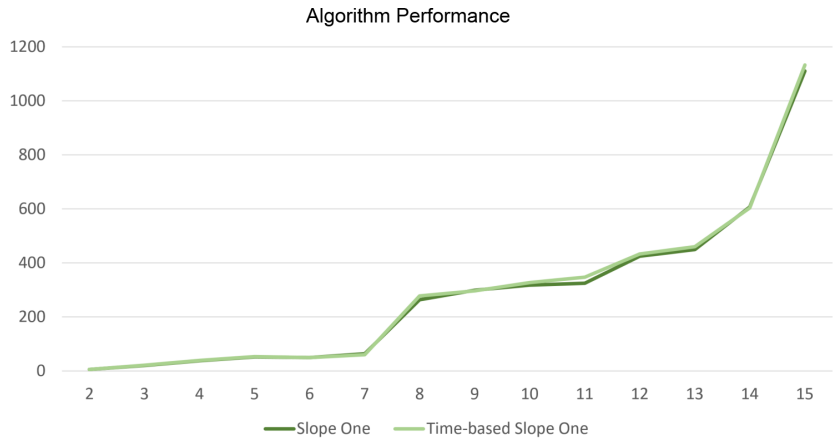

Figure B.27.: Slope One and Time-based Slope One performance comparison – average duration for the recommendation prediction given in seconds per user.

Table B.5.: Ratio of Content Repetitions in the *AWT* course

Week	Consumed Learning Units	Repeated Learning Units	Ratio of Repetitions
1	178	0	0%
2	223	87	39%
3	129	41	32%
4	148	58	39%
5	207	80	39%
6	135	29	21%
7	57	24	42%
8	114	34	30%
9	25	11	44%
10	27	16	59%
11	183	51	28%
12	275	77	28%
13	318	102	32%
14	794	264	33%
15	1089	407	37%
16	1889	878	46%

B.47. Presentation of SLR Recommendations in LCA

In the *Learning Companion Application*, the recommendations are presented in an overall Top-5 list with the title "You should learn the following topics" as shown in Figure B.28. As one can

see, in the beginning, the contents that have not been consumed so far or the ones with highest possible relevance are preferred by the algorithm. Besides this general Top-5 list, single context factor recommendations are also presented. These factors are presented to the learner as follows:

- Lecture Time: "You should prepare or wrap-up the following topics of the lecture"
- Collaborative Learning Need: "Classmates currently learn the following"
- Self-Assessments: "You have a low self-assessment in these topics"
- Exercises: "Your exercise results can be improved here"
- Interaction & Processing Time: "These are items you have learned insufficiently"
- Forgetting Effect: "You might have forgotten these *Learning Objects*"

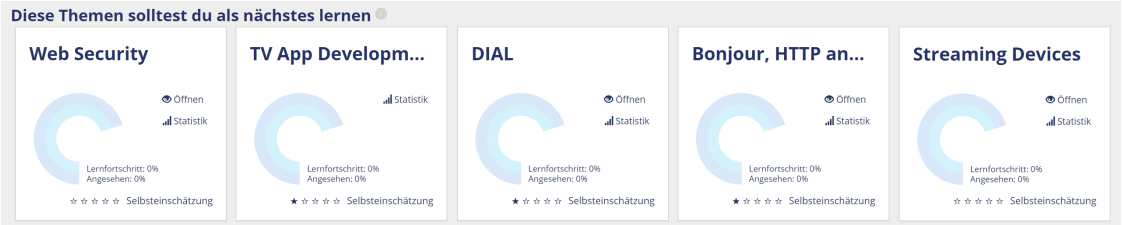

Figure B.28.: Example of Top-5 recommendations in the *LCA*

As shown in Figure B.28, the Top-5 list might comprise top-level items, such as "Web Security" or "TV App Development". Moreover, it might also contain low-level items, such as "DIAL", "Bonjour [...]" and "Streaming Devices" which belong to another top-level *Learning Object* ("Multiscreen Development"). The latter is not separately presented in the Top-N list to avoid redundancy. The reason, why child items of "Multiscreen Development" are preferred is that learners consumed other child items of the topic and, thus, they show a smaller learning need.

B.48. SLR Context Factor: Self-Assessments

Figure B.29.: Example of self-assessments in the statistic view in *LCA*.

Figure B.29 visualizes the statics view where learners can observe their activities and Figure B.30 presents the presented learning goals in a *Learning Unit*.

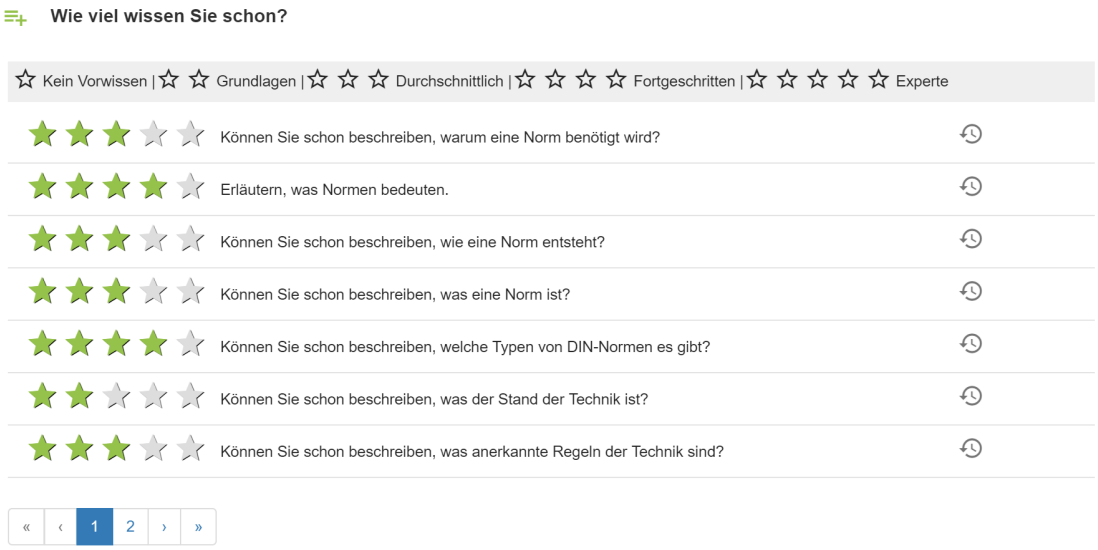

Figure B.30.: Example of self-assessments on learning goals at the beginning of a learning unit in *LCA*.

B.49. SLR Context Factor: Processing Time of an Item

As learners can, of course, be interrupted during learning or actively pause the knowledge acquisition, all learning sessions are summed and just compared to the total intended "instructional" time as shown in Figure B.31.

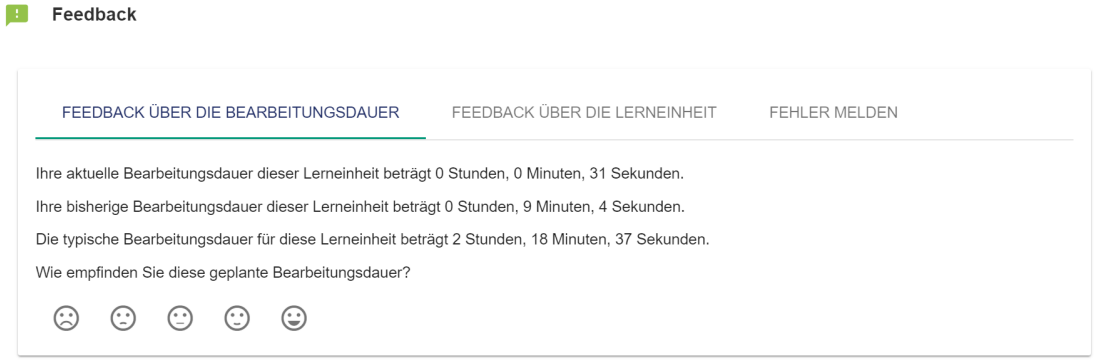

Figure B.31.: Example how the processing time is presented to the learner in *LCA*

In an initial phase the relevance score has been calculated as follows:

> "This factor indicates how long the student learned a *Learning Object*. It is 0 when the student needed exactly the intended time and between 0 and 1 if he [/ she] needs more or

less time than defined in the metadata.

$$rscore_{u,i,t,x3} = \left| 1 - \sqrt{\frac{timeNeededForLearning_{u,i,t}}{timeIntendedForLearning_i}} \right| \tag{B.7}$$

In an initial phase, [*SLR* works] with an upper bound for the time that is needed for learning: The square root lessens the effect when a student did not exactly learn the intended time. If the user needed more than 4 times of the time, the learning need is 1. In combination with the percentage of interaction, the processing time allows a good approximation whether a student really worked through that content"

Cf. [166][208].

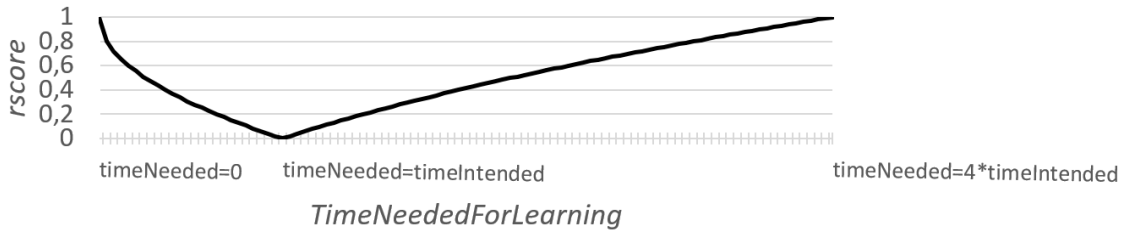

Figure B.32.: Range of relevance values for processing time

Figure B.32 visualizes the learning need based on the processing time that depends on the relation to the actual intended time. On the one hand, when a learner did not spend any time for learning, of course, the learning need is high. On the other hand, the first assumption was that learners usually do not spend more than four times the expected learning time. In that case, the learning need became high again. A user might open an item in *LCA* and then leave the working place while keeping the *Learning Object* open or the learner is otherwise distracted. In that case, the stopped engagement time would still increase, but the user does not acquire any knowledge. That is why the defined learning need was only low when engagement and instructional time are similar.

Feedback of course participants, however, indicate an important weakness of that approach. Learners want to reduce their visually presented learning need actively and, thus, want to optimize this factor as well. When learners start repeating contents, they expect the learning need values to decrease continuously, because the knowledge would strengthen due to the repetition. With the original equation, learners could not actively reduce the learning need, if they first spend more time than intended by the teacher. That is why the second half of the factor Equation B.7 is cut off and the first half is adapted to a linear model, where the learning need increases after spending more time than initially intended. Thus, when $timeNeededForLearning_{u,i} > timeIntendedForLearning_i$

[208] The paper excerpt has exclusively been written by Christopher Krauss.

the $rscore$ is set to 0.

B.50. Introduction of the Forgetting Effect

The following paper excerpts and texts introduce the forgetting model of Ebbinghaus [99] and how it is transferred to approximated forgetting of studied digital media. Thereby an experiment was conducted to identify influences of different parameters.

"Hermann Ebbinghaus [99] gave the first and still representative equation for the forgetting curve. He noticed that forgetting is high during the initial period after learning and gradually decreases over time. An experiment was conducted by Harry P. Bahrick [24] to test the recall and recognition of 50 English-Spanish word pairs over a period of 8 years. The results showed that the recall and recognition percentage of words is greater in larger intersession intervals, indicating the influence of spacing on forgetting. In recent years, some recommender systems re-used the equation from Ebbinghaus to improve predictions for the e-commerce and entertainment domain [129] [297].

[...] The decay theory [46] states that a person's memory of a learned content fades away over time, when it is not used. That is why forgetting represents a special relevance factor function $rf_{u,i,forgetting}(t)$ in the *Smart Learning Recommender*. Apart from time, we identified the following parameters that influence forgetting:

- Media type: The type of learning objects plays a vital role in remembering (cf. [232] [263]). In our approach, the media types are text, exercise, audio, graphics/image, video and multimedia.
- Difficulty level: The learning content represents different difficulty levels [263]. A common way of categorizing difficulty level is easy, medium and hard. It can be set based on the amount of content, detail level of information and so on. Forgetting will increase from difficulty level easy to hard.
- Prior knowledge: Learner's prior knowledge about the course helps to easily understand the course as compared to a learner who is new to the course. Hence, with prior knowledge, the learner remembers more [232] [263].
- Learner's interest towards the content: If the learner is not interested in a subject, forgetting tends to be at a higher rate compared to the subject of interest [263].
- Learner's memory strength: Every person in the world is different from each other, so their memory. A learner with higher memory strength can remember more, compared to a learner with lower memory strength.
- Repetition: During repetitions, students learn the offered items again. Repetition is

very helpful in strengthening the memory of learned content [99, 211].

- Repetition spacing: Constant repetitions at regular intervals will help to retain learned content. However, when the time spacing between initial learning and repetition is large, the percentage of increase in memory of the content is also high compared to the percentage increase with short repetition interval [211].
- Re-remembrance due to retention tests: Most research does not take the effect of retention tests for the re-remembrance into account: When people are asked about a topic, the questions in the retention test can act as a retrieval cue and allow the learner to correlate the words in the question to the previously learned content – thereby, remembering the forgotten concept as a side-effect.
- Retention test spacing: Similar to the repetition spacing, the gap in time between two successive retention tests can also have an influence on the re-remembrance. The more spacing, the smaller the effect of re-remembrance"

Cf. [166][209].

Factors like media type, difficulty level, repetition, learner's prior knowledge, learner's memory strength are studied by many researchers. This study includes previously examined factors as well as novel factors like learner's interest towards the content, re-remembrance due to retention tests and it's spacing concerning time.

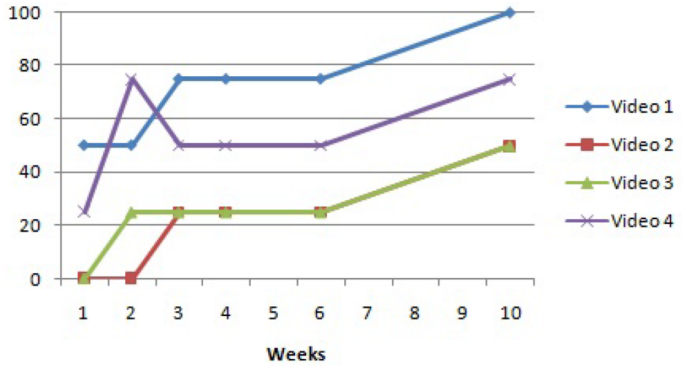

Figure B.33.: Survey results for forgetting of videos with different complexity levels that have only been watched at the beginning of the experiment. The y-axis represents the percentage of wrong or not answered questions per retention test. Thereby each 12.5% stand for one of eight answered questions.

"We conducted an experiment with a group of eight people to confirm and study the effect of the parameters that could influence forgetting. The duration was eight to ten weeks

[209] This paper excerpt has primarily been written by Rakesh Chandru and presents his work on the forgetting effect in his Master Thesis "Oblivion in Recommender Systems – The forgetting effect in predictions" which was supervised by Christopher Krauss.

and involved people from different fields like information technology, medicine and business administration. The eight people learn a learning object at the beginning of the experiment.

In regular intervals, we performed retention tests to observe the learner's knowledge. 5-8 questions from a group of 10-12 questions that represent the key information of the given topic are randomly picked and posed at the person to test the percentage of forgetting. The assumption: the progress of wrongly answered questions (in percent) over time represent the progress of forgetting. In the experiment, the parameter values are varied, to get a clue about its effect on forgetting. We presented different media types, in terms of texts and videos with different complexity levels. In some cases, the learning objects needed to be learned again at specific points in time to evaluate the repetition effect. Figure B.33 shows an excerpt from the survey results with the forgetting progress of videos that have been watched only once and show different lengths and complexity levels"

Cf. [166][210].

We evaluated the in the initial experiments determined formula in another study. Therefore, we compared the prediction accuracy of the actual Ebbinghaus model with the new generated model for the *Smart Learning Recommender*.

"Since this work shows a novel approach on forgetting and only a few data sets were published (e.g., [249]), which do not match all requirements of this approach, we generated our own data. We conducted a similar experiment as at the beginning in order to evaluate the correctness of our thesis. In our evaluation, 11 participants learned a previously unknown *Learning Object* just once. Over a period of eight to ten weeks, they had to answer retention tests in regular intervals. Each questionnaire consisted of four to eight questions and every single question was just asked once. The evaluation resulted in the analysis of the prediction accuracy using *Mean Absolute Error (MAE)* and *Root Mean Square Error (RMSE)*. See Figure B.34 for error values of different participants.

The results are compared with those from the equation of Hermann Ebbinghaus [99].

Table B.6 shows the average values of *MAE* and *RMSE* for the results from this study and from the Ebbinghaus equation for forgetting. It can be noticed that the average *MAE* and average *RMSE* are nearly 3 times lower in the model developed in the current study compared to the existing model from Ebbinghaus. This indicates a higher prediction accuracy of the model developed for forgetting in the current study, but still requires further experiments. We incorporate the forgetting factor as well as the other factors for a learning object and thus, predict its overall learning need for the given user"

[210] This paper excerpt has primarily been written by Rakesh Chandru and presents his work on the forgetting effect in his Master Thesis "Oblivion in Recommender Systems – The forgetting effect in predictions" which was supervised by Christopher Krauss.

Cf. [166][211].

Table B.6.: Error values for forgetting models

Equation	MAE	RMSE
by Hermann Ebbinghaus	0.3518	0.3896
SLR Forgetting	0.1245	0.1461

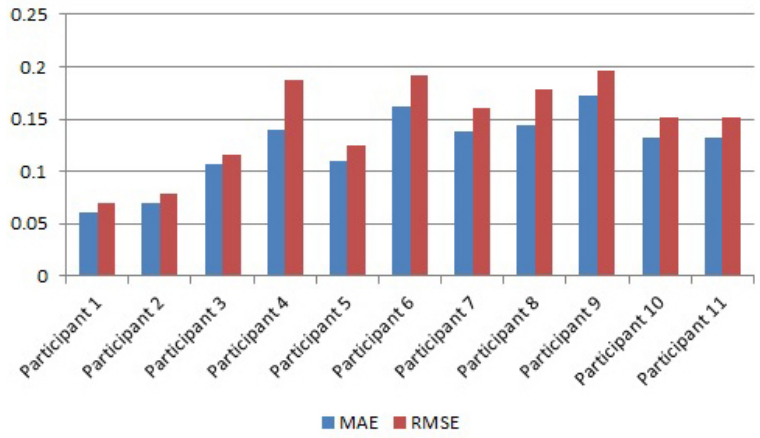

Figure B.34.: Error values of the predicted forgetting effect for different participants.

B.51. Explanation of the Forgetting Formula

The following paper excerpt presents details on the parameters of the forgetting model.

"[For the *SLR*, the] values for E_m are in the range $[0, 0.1]$. For example, media types which increase the retention by the highest possible value would be assigned 0. In our case, videos and animations are assigned 0.04 and text with 0.1. The difficulty level E_d can vary between 0 and 0.1, as well, where easy is assigned 0 and hard with 0.1.

[...] The equations for the effect of repetition and re-remembrance due to retention tests are given by Equations (B.8) and (B.9).

$$E_{ret} = n_{ret} * \frac{T_c - T_{d,ret}}{T_c} * K_{ret} \tag{B.8}$$

Where T_c is the total course duration expressed in days and $T_{d,ret}$ is the number of days after the last retention test. n_{ret} is the retention test count. The constant K_{ret} at the end of the equation indicates the weight by which the forgetting is reduced. The ideal value for

[211] This paper excerpt has primarily been written by Rakesh Chandru and presents his work on the forgetting effect in his Master Thesis "Oblivion in Recommender Systems – The forgetting effect in predictions" which was supervised by Christopher Krauss.

the constant is 0.01, based on the observations from the experiment.

$$E_{rep} = timeFactor * \frac{T_c - T_{d,rep}}{T_c} * K_{rep} \tag{B.9}$$

E_{rep} includes the *timeFactor* that was initially given by Formula (8.12), $T_{d,rep}$ is the number of days elapsed after last repetition. The constant K_{rep} at the end of the equation indicates the weight by which the forgetting is reduced. In our experiment the ideal value for this constant is 0.75.

[...] Factors like memory strength and the learner's interest towards the content are specific to each learner. These factors make the forgetting curve unique for each learner, but need further evaluations. The adapting constant α requires the conduction of retention tests. If there is no retention test planned, the adapting constant is set to 0. The value α personalizes the forgetting curve by taking the learner's performance in the retention into account. It is given by the following formula:

$$\alpha = \frac{predictedScore - actualScore}{2} \tag{B.10}$$

It represents the deviation between the regular forgetting curve [...] and the real forgetting progress of a single person – determined by retention tests"

Cf. [166][212].

B.52. History of Determining Appropriate SLR Weights with a Linear Model

Average Factor Values as Weights

For a more intelligent approach, the influence of the different factors in an equal weighted *SLR* setting is analyzed. Therefore, the generated factor values $rscore_{u,i,t,x}$ are averages for every item i in the training set (with max z items). The average values represent a weight $w_{x,u,t}$ for factor x, user u and the time of the recommendation t. The decay effect penalizes older data and prefers younger.

$$w_{x,u,t} = \frac{\sum_{i=0}^{z} \left(decay_{t,t_i} * rscore_{u,i,t,x}{}^{\beta}\right)}{\sum_{i=0}^{z}(decay_{t,t_i})} \tag{B.11}$$

$$decay_{t_i,t} = \frac{t_{first} - t_i}{t_{first} - t} \tag{B.12}$$

[212] This paper excerpt has primarily been written by Rakesh Chandru and presents his work on the forgetting effect in his Master Thesis "Oblivion in Recommender Systems – The forgetting effect in predictions" which was supervised by Christopher Krauss.

The decay $decay_{t_i,t}$ is higher, the closer the point in time of consumption is to the actual point in time of recommendation t. t_{first} is the first date on which an item access is considered for the weight (e.g., date of the course start) and t_i is the date where item i has been consumed[213]. The value β adjusts the power of the relevance value $rscore_{u,i,t,x}$ – the higher β, the more scattered are the single factor weights: High factor values get even higher impact (with a high value of β) and low factor values have a less effect.

Appendix B.53 presents the determined weights in detail. Interactions (determined weight of 15%)and the forgetting effect (determined weight of 15%) play minor roles in the weighting. Moreover, as mentioned previously the exam relevance is a constant and, at least in our *AWT* setting, equal for each item with a value of 1. Thus, it should not affect the item filtering process of the *Recommender System*, but receives the highest weight. That lead to the conclusion that the averaged factor values do not make much sense as factor weights.

Average Factor Deviation as Weights

For the third approach, the deviation of each item's relevance score from the average factor scores is determined. The intuitive approach behind that is that factor values with a high positive deviation from the average in the last days before the recommendation make the difference for appropriate recommendations and, thus, need to receive a higher weight. Based on the Formula B.11, the deviation Formula is adjusted as follows:

$$w_{x,u,t} = \frac{\sum_{i=0}^{z} \left(decay_{t,t_i} * \Delta rscore_{u,i,t,x}{}^{\beta} \right)}{\sum_{i=0}^{z} (decay_{t,t_i})} \tag{B.13}$$

Where $\Delta rscore_{u,i,t,x}$ is the deviation of the factor value $rscore_{u,i,t,x}$ from the average value $\varnothing rscore_{u,t,x}$ which can be expressed as

$$\Delta rscore_{u,i,t,x} = rscore_{u,i,t,x} - \varnothing rscore_{u,t,x} \tag{B.14}$$

Thereby, $\varnothing rscore_{u,t,x}$ stands for the average factor value for factor x of all items for user u at time of the recommendation t. $w_{x,u,t}$ must be greater than 0 to avoid negative weights. When $w_{x,u,t}$ has a negative value, it is set to 0 so that this factor does not influence at all. In other words: a factor is more important, the higher its deviation is from the average values of that factor.

A detailed analysis of this approach is presented in Appendix B.54. This extension shows, unfortunately, an even worse effect on the precision resulting in only 0.377% on average (a reduction of almost 20%). It turns out that the relevance scores of a particular factors are typically in sub ranges smaller than [0,1]. The average value of all exercise relevance scores, for instance, is 0.12, while the average interaction score is 0.30, the average processing time score is 0.35 and the

[213] By the way, this formula also works in the other direction, for item analysis in the test set. Nevertheless, to generate reliable results and strictly follow the guidelines of the evaluation framework, just items of the training set are considered.

average self-assessment score is 0.46.

B.53. Factor Weights of the Average Factor Values Approach

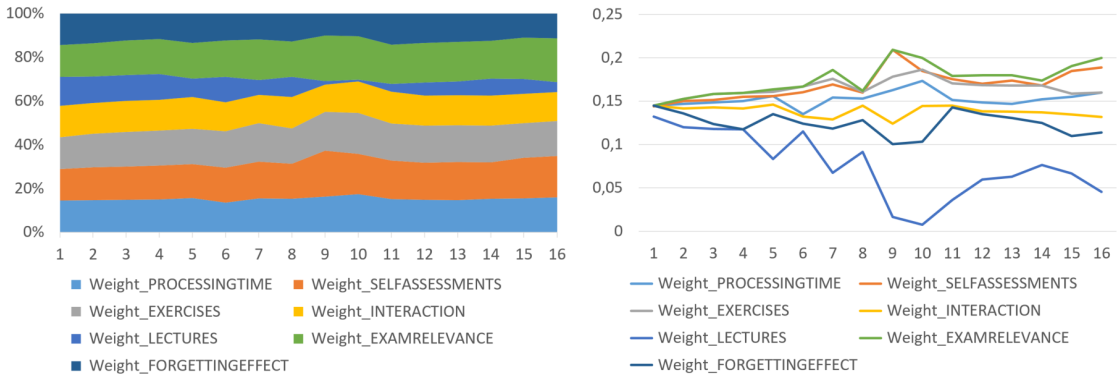

Figure B.35.: Importance of the factors over the course period in AWT as ratio of determined factor weights according to the average factor values approach (for Top-3 recommendations, $t_{first} = t - 7days$ and $\beta = 1$). left: values as stacked area; right: values as line chart

Figure B.35 shows two times the same ratio distribution of the factor weights: on the left for an overview of all weights and on the right side for a direct comparison of the relative weights of the analyzed factors in the AWT course. t_{first} was set to 7 days before the recommendation, which means that items consumed at this point have the lowest effect and items consumed at the time of the recommendation t show the highest effect on the weight. The power β is set to 1. As one can see, almost all factor weights start at the same initial weight of about 15% at the course start. Self-assessments, exercises, and exam relevance have an almost equal influence between 15 and 20% each. Processing time has less influence with about 15% over the whole period and interactions even less with 13% on average over the course period. The forgetting effect ranges between 10 and 15% influence.

Interestingly, the impact of lecture times reduces over the course period – starting with 14% and reducing to under 5% with a local valley of about 1% in weeks 9 and 10. The local valley is the Christmas time when there was no lecture at all. The overall reduction of the factor influence might be because the course schedule allows a differentiation during the lecture times, especially when users start following the course contents at the beginning but shortly after that follow their individual learning agenda.

Figure B.36 shows the same impact values of the factors grouped for all items that have not been consumed previously (left) and items that are repeated by the user (right). As one can see, the exam relevance, self-assessments and partly the lecture times are more important for items that have been consumed previously. It is undeniable for repeated topics that the lecture time

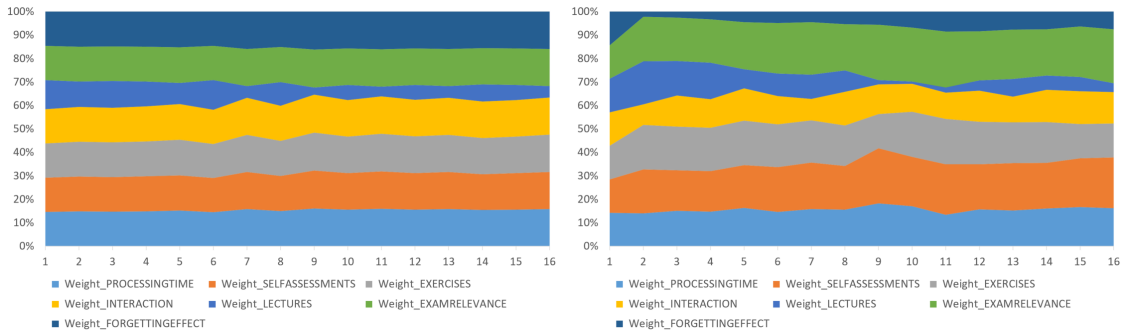

Figure B.36.: Detailed analysis of importance of the factors over the course period in AWT as ratio of determined factor weights according to the average factor values approach (for Top-3 recommendations, $t_{first} = t_r - 7days$ and $\beta = 1$); left: for all new items (that have been consumed for the first time); right: for all repeated items (that have been consumed previously)

factor influences the recommendations in the first half of the course and starting with week 11 again (after New Year's Eve) – only those course periods comprised face-to-face lectures.

B.54. Factor Weights of the Average Factor Deviation Approach

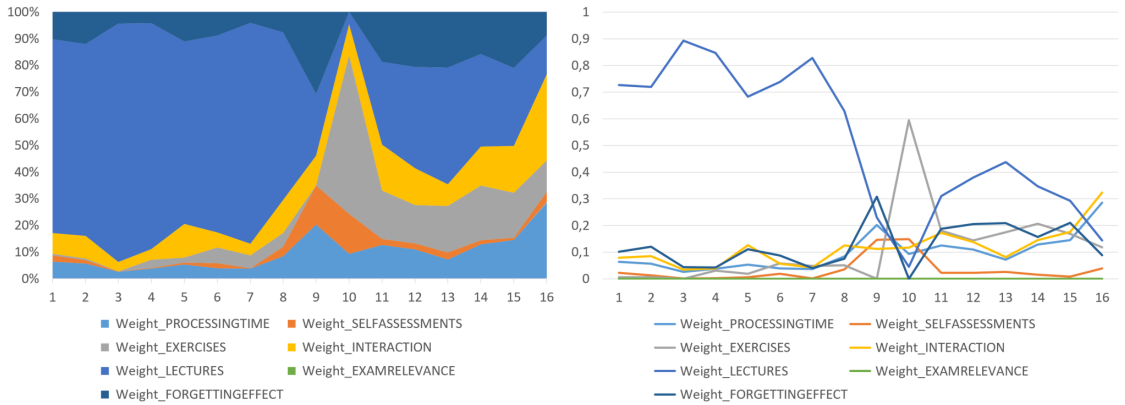

Figure B.37.: Importance of the factors over the course period in AWT as ratio of determined factor weights according to the average factor deviation approach (for Top-3 recommendations, $t_{relevant} = t_r + 7days$ and $x = 1$). left: values as stacked area; right: values as line chart

Figure B.37 shows the influence of the factor values due to the positive average deviation values. Thereby, lecture times play a major role in the recommendation selection process – except the Christmas Holidays. During this free period, in turn, self-assessments and especially exercise results indicate which items are particular of interest. Interactions and processing time show a similar progress with slowly increasing influence. As a conclusion, there is not much difference

between those two factors, as one represents a number of consumed sub-items and the other represents the time spend on an item – both have a similar influence on the total consumption. The constant value for exam relevance receives, as expected, a weight of zero because this factor does not influence the item consumption behavior at all.

B.55. Qualitative Study on LCA in AWT

In order to evaluate the subjective perception of recommendation effectiveness and efficiency, a qualitative study has been conducted: 29 participants of the *AWT* course[214] participated additionally to the course in a Thinking-Aloud Study. The participants performed seven tasks with the help of the *Learning Companion Application* and, thereby, were requested to speak loudly about their user experience and faced issues. Most tasks allowed different strategies to solve the task – e.g., open the contents of the last lecture via the course overview, the schedule, the search engine or the list of recommendations. Note: the external conditions (in terms of questioners and additional support during the tasks) are not well defined and, thus, the results are only partially meaningful as different test persons encountered different conditions. A new study with a formal definition of the experimental set-up is in progress, but could not be completed before the submission of the dissertation. The tasks can be summarized as:

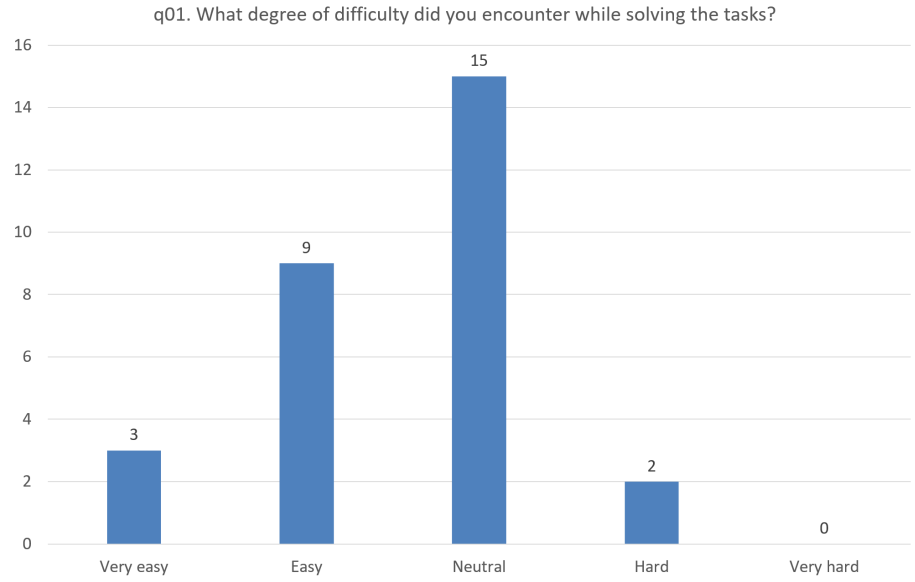

Figure B.38.: Survey question on the general difficulty after the thinking aloud session for the AWT course

- Download last week's lecture slides (Via recommendations or course overview)

[214]The participants comprise seven female, 21 male students. One did not provide feedback.

- Find learning contents presented in the next lecture (Via recommendations or course overview)
- Find learning contents for ”filtering approaches” (Via search or course overview)
- Find learning contents for ”HbbTV” (Via search or course overview)
- Identify your lowest self-assessment (Via recommendations or course overview)
- Find recommendations for contents you might have forgotten (Only via recommendations)
- Find the schedule for next week (Via search or course overview)

It took the participants between 4 min 18 sec and 22 min 17 sec to complete all tasks. While the utilization of the search engine and the page of the recommendations are similarly efficient in terms of needed time to solve the task (on average about 12 seconds), the recommendation list was only used in 10% of the cases and the search engine in 79%. Thus, recommendations seem to be helpful when users do not know what they are looking for (otherwise they would use the search feature). However, all participants opened the recommendation list at least once, and 83% (24 in total) of the participants appreciated the provided recommendations in the related task. The other five students (17%) argued about a missing English translation as the main reason for the encountered problems[215] that also comprises the German-only titles of the recommendations.

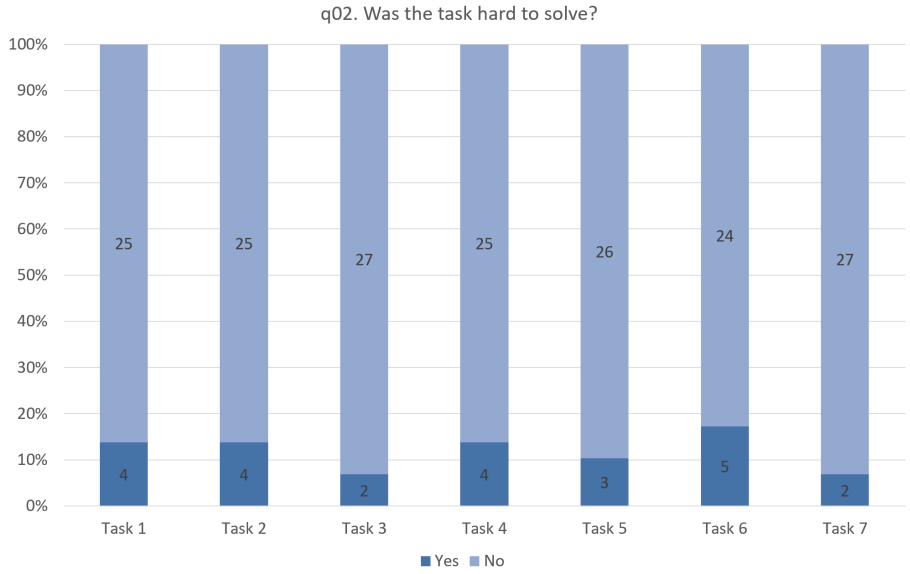

Figure B.39.: Survey question on the difficulty per task after the thinking aloud session for the AWT course

5 participants (17%) encounter problems when searching for a search button, that was expected to be in the top bar, but it was part of the main menu. 4 participants wished more interactive learning contents instead of slides and the navigation between contents is too complicated. And 3

[215] At the time of this study, the *LCA* was offered in the German language only.

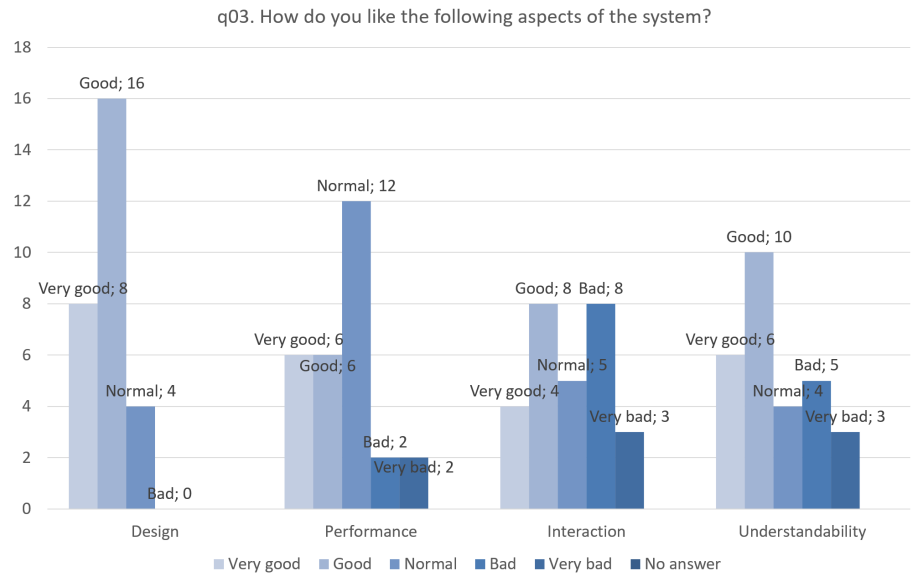

Figure B.40.: Survey question on the preference for specific aspects after the thinking aloud session for the AWT course

participants wanted a discussion forum as part of *LCA*.

After the thinking aloud studies, the participants are asked to answer an online survey that is presented in Figure B.38 to Figure B.40.

The same 29 participants are asked for a comparison of the *Learning Companion Application* and *Information System for Instructors and Students (ISIS)* which is the *TU Berlin* branded version of Moodle used for all course offerings. Only 15 students (out of 29) have completed an *ISIS* course so far. That implies that about the half of the participants are newly matriculated at *TU Berlin*.

21 participants find the navigation of *LCA* easy, while only 13 find *ISIS* easy regarding navigation. If they had the choice, 13 participants would decide for *LCA* and 13 for *ISIS*, the rest could not decide. The 6 of the 13 students who preferred *ISIS* give as a reason the missing English language of the *Learning Companion Application* (which is going to be integrated in January 2018).

B.56. Qualitative Study on LCA at Chamber of Crafts

The eight participants of the first energy consultant training with *LCA* at the chamber of crafts also participated in an online study. While the course was hosted at the Chamber of Crafts Berlin, the *IZT* surveyed with aggregated questions of the partners.

Six of the eight participants prepare and wrap-up the lectures. The time spend is almost equal for preparation and wrap-up and lies between 30 and 60 minutes per week. On the question, what the participants wish for future *Learning Management Systems*, all answered a digital lecture schedule and access to course materials. Seven of eight also wished recommendations for the

preparation of the final exam. Six participants want additionally an overview of their personal learning progress for each *Learning Object* and four each plead for recommendations for *Life-Long Learning* and for lecture preparation and wrap-up. Only two participants wished recommendations of peers to build learning groups.

On the question about the usefulness of a visualization of the learning progress, six participants answered with "very useful" and the other two with "useful". Moreover, all participants answered yes on the question, if they want to get hints on assumed learning weaknesses.

The *IZT* also asked open questions, such as "In your opinion, which advantages show *LMSs*?". Answers are "Better and deeper learning", "the personal mentoring", "flexibility", "Every-time access" and "Uncover learning weaknesses respectively deepen learning to counteract on them". On the question "Which doubts do you have on the usage of a learning recommender system?", the participants answered: "Disregard of data protection", "Monitoring" and "As long as it stays anonymous and nothing is forwarded to third parties, I have no doubts".

Statement of Affirmation

I hereby declare that this dissertation has been written only by the undersigned and without any assistance from third parties.
Furthermore, I confirm that no sources have been used in the preparation of this dissertation other than those indicated in the dissertation itself.