Modeling and Analyzing Bias in Recommender Systems from Multi-views: Context, Topic and Evaluation

vorgelegt von
M. Sc.
Jing Yuan

an der Fakultät IV - Elektrotechnik und Informatik
der Technische Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
-Dr.-Ing.-

genehmigte Dissertation

Tag der wissenschaftlichen Aussprache: 11. Mai 2021

Berlin 2021

What I can not create,
I do not understand.

– RICHARD FEYNMAN

Die Luft der Freiheit weht.

– ULRICH VON HUTTEN &
DAVID STARR JORDAN

ABSTRACT

With the explosive growth of information on the Internet, recommender systems have been broadly applied in user engaged systems to efficiently discover items of potential interest. In order to automatically generate such "guess what you like" results and serve matching recommendations, advanced machine learning and data mining techniques are applied in recommender systems. However, the model based methods that learn from historical data often result in unavoidable bias in recommendations from different perspectives. The *filtering bubble* reflected in the biased recommendations leads to the fact that recommended targets fall into a narrow range.

The existing research towards the bias problem in recommender systems focuses mainly on the bias adjustment within a specific modeling phase. The comprehensive understanding and generic bias countering approaches are still missing. In this thesis, we research on the bias problem in recommender systems from multi-views, including contextual bias, content-level understanding of bias, and the evaluation bias. These bias phenomena are observed in specific application scenarios. The modeling, analysis and evaluation of bias are conducted accordingly.

First, in the recommendation scenario of IPTV systems, recommendations are more sensitive to the contextual influence due to the video genres and airing schedule. Thus we conduct the research on modeling and countering contextual bias in this scenario. Second, digital news portals form a special recommendation scenario. The user impression or clicking behavior is heavily affected by the content-level bias understanding. Targeting on the gap between the article level popularity bias and the content-level understanding of bias, we research on the topical bias representation for news articles and their potential predicting power. In addition, in the algorithm recommendation scenario, the single objective evaluation leads to the overlook on the other measurement targets. Therefore, multi-objective evaluation and candidate expansion are attempted to deal with such evaluation bias. The proposed modeling approach in algorithm selection has proven to balance the multi-objective evaluation well.

ZUSAMMENFASSUNG

Mit den explosiven Informationen im Internet wurden Empfehlungssysteme weitgehend in benutzerbezogenen Systemen eingesetzt, um potenzielle interessierte Elemente schnell zu entdecken. Um solche Ergebnisse automatisch zu generieren und entsprechende Empfehlungen abzugeben, wurden in den Empfehlungssystemen fortgeschrittene Techniken für maschinelles Lernen und Data Mining versucht. Die modellbasierten Methoden, die aus den historischen Daten lernen, führen jedoch zu unvermeidlichen voreingenommenen Empfehlungen aus verschiedenen Perspektiven. Die *Filterblase*, die sich in der voreingenommenen Empfehlung widerspiegelt, führt dazu, dass empfohlene Ziele in einen engen Bereich fallen.

Die vorhandenen Untersuchungen zum Bias-Problem im Empfehlungssystem konzentrieren sich hauptsächlich auf die Bias-Anpassung innerhalb einer bestimmten Modellierungsphase. Das umfassende Verständnis und der generische Bias-Countering-Ansatz fehlen noch. In dieser Arbeit untersuchen wir das Verzerrungsproblem im Empfehlungssystem aus mehreren Ansichten, einschließlich kontextbezogener Verzerrung, Verständnis der Verzerrung auf Inhaltsebene und der Bewertungsverzerrung. Diese Verzerrungsphänomene wurden in bestimmten Anwendungsszenarien erfasst. Die Modellierung, Analyse und Bewertung des Bias wurde entsprechend durchgeführt.

Zuerst, das IPTV-System reagiert aufgrund der Videogattungen und des Sendeplans empfindlicher auf den kontextuellen Einfluss. In diesem Szenario werden daher Untersuchungen zur Modellierung und Bekämpfung kontextbezogener Verzerrungen durchgeführt. Zweite, in digitalen Nachrichtenportalen wird das Eindrucks- oder Klickverhalten des Benutzers stark vom Verständnis der Verzerrung auf Inhaltsebene beeinflusst. Ausrichtung auf die Lücke zwischen der Beliebtheit der Artikelebene und dem Verständnis der Verzerrung auf Inhaltsebene. Die Analyse wurde in Richtung der aktuellen Bias-Darstellung für Nachrichtenartikel erstellt. Darüber hinaus kann das im Empfehlungssystem verwendete Einzelobjektbewertungsverfahren zu einem voreingenommenen Ziel führen. Bei der Bewertung mehrerer Ziele und der Erweiterung der Kandidaten wurde versucht, die Bewertungsverzerrung zu bewältigen. Der vorgeschlagene Modellierungsansatz hat Bewertung mit mehreren Zielen gut ausbalanciert bei der Algorithmusauswahl.

ACKNOWLEDGMENTS

At the accomplishment phase of my dissertation, I would like to represent my gratitude to many people, without whom the thesis is impossible to be finished.

First and foremost, I would like to appreciate Prof. Dr. Sahin Albayrak, who gave me the opportunity to continue the study and research on my Ph.D. thesis in TU Berlin. Meanwhile I would also like to thank Prof. Dr. Arkady Zaslavsky and Dr. Frank Hopfgartner for being on my thesis review committee.

The nice colleagues at DAI-Labor and GT-ARC at TU Berlin are the ones I need to thank most. Special thanks go to Dr. Fikret Sivrikaya, who introduced me to the Lab and enlightened my research; Dr. Andreas Lommatzsch, who provided me endless encouragement and opportunities to do the research and teaching; Dr. Frank Hopfgartner, who guided and supported me at many key turning points; Dr. Brijnesh Jain and Dr. Till Plumbaum who shed the light towards the final chapter of the thesis. I spent the first 4 years of research time in CC-NEMO, where I need to thank Dr. Manzoor Ahmed Khan, Sebastian Peters, Thomas Geithner, Stefan Marx, Nadim EI Said, Carsten Niebuhr, and Xuan-Thuy Dang for the precious friendship and the project co-working. In the later time working in the CC-IRML, many thanks go to Benjamin Kille, Christian Geissler, Weijia Shao, Danuta Ploch, Christian Sauer and Michael Meder for the knowledge sharing and the research discussion.

In addition, I need to express my gratitude to the co-authors of the publications: Christian Geissler and Weijia Shao, without your valuable advices, the final publication can't be realized. Benjamin Kille, thanks for your patient proof reading and revision, they set me good examples concerning the scientific writing. Felix Lorenz, a publication together with you from a seminar to project during teaching is wonderful. Dr. Mu Mu and Dr. Nicholas Race, the data acquisition support from your side means a lot to me.

Very importantly, I must sincerely thank Chinese Scholarship Council (CSC) for the financial support during my Ph.D. studies.

Last but not least, my deepest gratitude goes to all my lovely family members. Especially my mother who sets me the good example of never giving up. Hearty appreciation goes to my dear husband, who strongly supported me went through all the difficult moments.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Recommender System (RS) provides users with quick access to massive information resources according to their personal preference. The applications of RS span across from e-commerce systems to specific individualized systems (e.g., algorithm selection). However, recommenders are easily biased towards the skewed items, because of either the training data or the chosen evaluation metrics. Such bias problems lead to potential degradation in the quality of an RS. In this thesis, we model and analyze the bias in RS from multi-views. The contextual bias in IP-based Television (IPTV) systems, the content-level understanding of popularity bias and evaluation bias in the algorithm selection scenarios are studied separately. In this introduction chapter, Section 1.1 describes the motivation and goals of the thesis, after which Section 1.2 illustrates the scenarios and scope of the thesis. Subsequently, the challenges of the thesis are introduced in Section 1.3. The contributions and relevant publications are listed in Section 1.4. Finally, the structure of the thesis is introduced in Section 1.5.

1.1 Motivations and Goals

Since emerged in the 1990s, RSs have been successfully applied in various online commercial systems. Suggestions on what to buy, what to watch, even whom to date are tailored according to users' individual preference by recommenders [1]. RS thus brings in a win-win situation for e-commerce, where users acquire what they like more conveniently, while commercial providers benefit from higher user stickiness. In the year 2006, owing to the fierce competition on Netflix prize[1], RS attracted extensive attention from the research community and became active in both laboratory environment and real-life applications [1–3]. Various Data Mining (DM) and Machine Learning (ML) algorithms have been tried in RS to score and rank items for specific individuals.

[1]https://www.netflixprize.com/

An RS problem can be represented as a mapping function $f : u, i, c \mapsto \mathbb{R}$ [4]. Here, user factor u represents a recommender's serving object. Depending on the concrete scenario, u denotes either plain individual indicator or a profile with plentiful meta-features. The item factor i stands for the candidate object, it serves either as a standalone indicator or includes the descriptive features of the candidate items. Context factor c represents the combination of multiple context information, like *time of day*, *day of week*, *location*, etc. Preference value $\mathbb{R}$ is the accordance for ranking the candidate items for users. It may refer to a rating scale, a probability value, a frequency prediction, etc. To solve a recommendation problem, the historical data in a specific scenario is the source for building the learning models, and the pre-defined evaluation metric is the optimization target.

It has been broadly accepted in the research community that increasing the recommendation accuracy is the main goal of an RS. The algorithms that predict the user ratings more precisely or attract higher Click Through Rate (CTR) are more favorable in a recommender. Thus the accuracy or prediction error oriented evaluation becomes the optimization purpose in recommender scenarios. However, historical datasets contain many bias problems. For instance, if a user has watched several action movies from an online video platform, RS captures this fact and continuously presents action-like genre movies to users. It thereby increases the chance that the user chooses an action movie again, and reduces the likelihood that the user watches other potentially interesting video items from other genres. To overcome this issue, the trade-off between recommendation accuracy and de-biasing effect become a hot topic in the research community in recent years. There are different perspectives to understand the bias phenomenon in an RS [5–9]. We summarize them as the following:

- *Users' behavioral bias* results from the users' behavioral habits. On one hand, users have their own habitual selection on the representation form of the chosen items. For instance, items' ranking positions, representation layout, triggering button clicking position, etc. [10, 11]. On the other hand, *users' behavioral bias* is also reflected by users' individual preferred choices on specific items. To counter such bias from the *implicit feedback*, specific weighting schema in the modeling phase needs to be considered. Understanding users' preference pattern also helps understanding the *users' behavioral bias*.

- *Popularity bias* on items heavily influence users' choices in a recommender system. The dominant items make the historical data skewed. Influenced by such skewed data, RS inclines to select the popular items thus creates the *filtering bubble*. Though personalization is realized in the usual RS, the popularity embedded in the user interactions still makes the bias unavoidable [7]. The popularity bias is tractable on the item level in a straightforward view, the literal understanding of it on a fine-grained level still represents the research gap.

- *Evaluation bias* in a recommender is recognized as the limited optimization target while learning a recommendation model. To evaluate the effectiveness of an RS system, prediction error compared with the ground truth ratings and the accuracy of the $TOPK$ items in the recommendation list are the usual metrics. Intuitively, the better the recommender predicts the ratings of specific users, the more the user's individual preference is presented. However, these metrics do not cover all the perspectives concerning users' and systems' satisfaction. Other metrics like response time, diversity and novelty are all important factors to be considered [12]. The calibration on the evaluation bias is needed when the recommender is expected to behave well on other factors as well.

Facing the multi-views of the bias problem in RS, the goal of this thesis is to model and understand the bias in concrete RS scenarios, find ways to counter or utilize such bias for constructing better recommenders. To achieve this goal, we conduct the studies within the following scenarios and scope.

1.2 Scenarios and Scope

Recommendation targets alter when the recommendation scenarios change. Typical recommendation targets and recommendation scenario combinations include: commodities in online shopping websites like Amazon[2], videos streamed from online video providers, e.g. Youtube[3], articles created by online news portals such as Tagesspiegel[4], mate candidates on dating platform for instance Lovoo[5]. Aside from these e-commercial scenarios, RSs are also applicable in other static knowledge relevant domains. In the scenario of Algorithm Selection (AS), algorithm candidates are recommended to solve a specific problem instance. OpenML [6] is a typical example for the performance collection of algorithm candidates on specific problem instances.

The modeling and analysis of the bias in both e-commercial systems and static knowledge systems deserve the attention and study. The recommendation strategy needs to be designed accordingly for the specific characteristics in a recommendation scenario. For instance, the literal understanding of the article content is important in digital news portals, whereas the avatars gaining more attraction weigh an item profile higher in an online

[2]https://www.amazon.com
[3]https://www.youtube.com
[4]https://www.tagesspiegel.de/
[5]https://www.lovoo.com/
[6]https://www.openml.org

dating website. Or in non-user-oriented scenarios like algorithm selection, the meta features for u are dense rather than sparse (in e-commercial systems the meta features of u are usually quite sparse), since they are generated by the same standard process. While designing a recommender and analyzing the bias issues, the scenario characteristics need to be distinguished and treated differently.

In this dissertation, we mainly focus on the bias problems in three recommendation scenarios; namely, IPTV program recommendation, digital news article recommendation and algorithm recommendation. The traits of these scenarios and the research scope in the thesis are listed in Table 1.1 accordingly.

Table 1.1: Recommendation scenarios covered in the thesis.

Recommendation Scenarios	Characteristics	Research Scope
IPTV Systems	• Airing schedule of TV programs is regular. • Personal watching habit varies individually. • *Contextual factors* matter for catching users' preference pattern.	• The role of contextual factors(user study); • modeling of contextual factors (context-aware Latent Dirichlet Allocation); • countering contextual bias via involving external sources.
Digital News Portals	• Texts are the main content to be consumed. • Hotly discussed breaking events heavily influence the articles' popularity.	• Clicks pattern analysis for the popular articles; • content level ranking bias tracking; • topic ranking bias tracking from temporal and behavioral view.
Algorithm Selection	• Multi-model or one-hot conversion are needed in conventional approaches. • Pure accuracy oriented metrics could cause evaluation bias. • *TOP1* candidates selection in evaluation can bias the choices.	• Introducing Bi-linear Learning to Rank; • using A3R as multi-objective evaluation strategy; • *TOPK* expansion in the candidates set in the evaluation.

1.3 Challenges

In order to model and analyze the bias problems in recommender systems, there are several challenges to be dealt with.

- The bias modeling in the recommender increases the parameter space and the complexity of the model, thus is prone to be over-fitting on the skewed dataset. The reliance on system level internal user data source makes it difficult to escape from the notorious *filter bubble* in the recommender. Getting rid of the reliance on the system-internal data is a challenging task.
- The reasonable explanation of the bias is obscure in most recommenders, especially on the topic or content level understanding. With the parsed biased topic, how to measure the bias degree and utilize such bias degree ranking to boost the prediction power brings in challenges.
- To achieve a balanced evaluation protocol, it is challenging to incorporate multiple metrics into a single representation. Especially when the metrics are in different ordering favors, a fair combination is difficult to realize. Meanwhile, a recommender algorithm which is supposed to properly adapt to the combined metric needs a careful design.

The research conducted in this thesis focuses on tackling these challenges and designing relevant solutions to overcome them in specific recommendation scenarios.

1.4 Contributions and Publications

Main contributions of the thesis are summarized in this section, together with a list of the relevant publications by the author [13–19].

Contextual bias modeling and countering in the IPTV scenario. As discussed in the above sections, contextual variations are obvious in the IPTV systems, thus the contextual bias is of research interest. First we study the role of contextual factors in a recommender. A user study towards the contextual influence was conducted, where the survey feedback explains the context influence in an IPTV system. Subsequently, with the conclusion of the user study, a context-aware Latent Dirichlet Allocation (LDA) recommendation algorithm has been proposed, which integrates the contextual influence and users' inclination on the recommender path into the model. The newly created context-aware LDA performs well on both precision and diversity oriented evaluation metrics. In the end, to counter the contextual bias in the IPTV recommender, the domain relevant topic trend has been captured from Twitter. It serves well as a supplement recommender especially for the purpose of de-biasing users' contextual skewed choices. The relevant publications concerning this contribution are as follows.

- **Jing Yuan**, Fikret Sivrikaya, Stefan Marx, Frank Hopfgartner, "When to Recommend What? A Study on the Role of Contextual Factors in IP-Based TV Services", Workshop Mind the Gap, Berlin, Germany, 4th March 2014.
- **Jing Yuan**, "How Adaptive Aggregate Contextual Recommender Benefits IP-based TV Services", Doctoral Consortium in ACM International Conference on TVX, Newcastle upon Tyne, UK, 26th-27th June 2014.
- **Jing Yuan**, Fikret Sivrikaya, Frank Hopfgartner, Andreas Lommatzsch and Mu Mu, "Context-aware LDA: Balancing Relevance and Diversity in TV Content Recommenders", workshop RecSysTV in conjunction with RecSys, Vienna Austria, 16th-20th September 2015.
- **Jing Yuan**, Andreas Lommatzsch and Mu Mu, "Applying Topic Model in Context-Aware TV Programs Recommendation", track Knowledge Discovery, Data Mining and Machine Learning in LWDA, Potsdam, 12th-14th September 2016.
- Felix Lorenz, **Jing Yuan**, Andreas Lommatzsch, Mu Mu, Nicholas Race, Frank Hopfgartner, Sahin Albayrak, "Countering Contextual Bias in TV Watching Behavior: Introducing Social Trend as External Contextual Factor in TV Recommenders", ACM TVX2017, Hilversum, the Netherlands, 14th-16th June, 2017.

Explainable popularity bias from content-level in digital news recommendation scenario. In systems with implicit feedback, bias is reflected by the high click-through rates of the most popular items. A vast space of the items are ignored and will not be raised again in the system [6]. Understanding the bias from content level is very important to track the reason of the popularity bias. In order to understand the popularity bias, the selection frequency on the item level is usually considered as the bias representation. The weighted item frequency is converted to the topic ranking bias. Such topic ranking bias is tracked from both the temporal view and between different behaviors. The textual parsing and the descriptions extractions are developed, which help represent the prediction power in the news recommender. Below are the relevant publications and technical reports in this context:

- **Jing Yuan**, Andreas Lommatzsch, Benjamin Kille, "Clicks Pattern Analysis for Online News Recommendation Systems", NEWSREEL (real-time news recommending challenge) in CLEF, Portugal, 5th-8th September 2016.

- **Jing Yuan**, Andreas Lommatzsch, Fikret Sivrikaya, “Analysis on Users’ Topic Concern Bias in News Stream”, technical report, 2017.

Multi-objective evaluation metric and a novel approach for algorithm recommendation. When it comes to the algorithm selection scenario, the research goal is to balance the evaluation for both prediction and inference time. A new evaluation metric Adjusted Ratio of Root Ratios (A3R) is proposed to represent such balance. For adapting to the new metric, algorithm Bi-Linear Learning to Rank (BLR) has been proposed. BLR overcomes the drawbacks in the conventional solutions like multi-model building and one-hot encoding. It has proven to outperform in the selection effect especially concerning the trade-off evaluation for both accuracy and inference time. In addition, we affirm the benefit of expanding the selection range of candidate approaches from $TOP1$ to $TOPK$ regarding the cumulative optimal demand of AS evaluation. Such expansion helps the selection effect get rid of the biased $TOP1$ selection limitation. The following publications form this contribution.

- **Jing Yuan**, Andreas Lommatzsch and Sahin Albayrak, “Contextual Factors Involvement for Score Modeling under LambdaRank in Recommender Systems”(long abstract), workshop Women in Machine Learning, co-located with NIPS, Barcelona, Spain, 4th-5th December 2016.
- **Jing Yuan**, Christian Geißler, Weijia Shao, Andreas Lommatzsch, Brijnesh Jain, “When Algorithm Selection Meets Bi-linear Learning to Rank: Accuracy and Inference Time Trade Off with Candidates Expansion”, International Journal of Data Science and Analytics, published on 9th Oct. 2020.

1.5 Structure of the Thesis

This thesis is structured as follows. Chapter 2 introduces the background knowledge about recommender systems and the bias phenomenon in different RS scenarios. Chapter 3 raises the concept of contextual bias in TV recommender systems, and proposes modeling contextual factors under the LDA probablistic framework and involving domain social trend to tackle the bias issue. Moreover the contextual bias degree and the bias-countering effect have been measured and analyzed in the way of grid entropy. In Chapter 4, the topic concern bias in a news recommender system is extracted from content level instead of item level. The measurements are designed from temporal and behavioral views to quantify the bias differently. Thereafter, given the scenario of algorithm recommendation, Chapter 5 proposes the Bi-linear Learning to Rank to solve the algorithm ranking and recommendation problem in the AS domain. The proposed approach performs especially well when balancing the precision biased evaluation and the inference time with the evaluation metric A3R. The accumulated $TOPK$ evaluation effect is verified as the countering factor of the $TOP1$ selection evaluation bias. Finally, Chapter 6 concludes the thesis and presents an outlook to the future research directions under the topic *bias modeling and analysis in recommender systems*.

CHAPTER 2

BACKGROUND

In this Chapter, we introduce the research background of Recommender System (RS) and the bias issues in different scenarios. The literature reviews are given from multi-views. First, Section 2.1 describes the classic abstract level understanding of RS from *user*, *item* and *context* perspectives. The selection bias and serendipity are summarized accordingly. Second, focusing on the content level understanding of an RS, the content parsing techniques and the popularity bias analysis are specified in Section 2.2. Subsequently, Section 2.3 concerns the algorithm recommendation scenario and evaluation protocols in the RS. The literature overview of both algorithm selection and evaluation approaches raise the thought of evaluation bias in the RS. Finally, Section 2.4 summarizes the chapter with a short conclusion.

2.1 User, Item, Context: Selection Bias and Serendipity

When interpreting the essence of an RS from an abstracted level, an RS problem can be represented in a mapping relation $f : u, i \mapsto \mathbb{R}$, where u stands for a user, i denotes an item, and $\mathbb{R}$ indicates the preference degree user u has on item i, i.e. User Rating Matrix (URM). The entries of matrix $\mathbb{R}$ can be a rating scale, a probability value scaled in $[0, 1]$, a frequency value or a launch duration etc. [20–23].

A proper mapping function gives precise prediction on $\mathbb{R}$. If no extra meta information is provided concerning u and i, the mapping relation is learned solely from $\mathbb{R}$. Once user's profile information (like age, interest group, location, gender) or item's features (like size, color, brand, single etc.) are also accessible, the recommendation algorithm needs to take the three aspects information: URM $\mathbb{R}$, user feature matrix $\mathbb{U}$ and item feature matrix $\mathbb{I}$ into consideration. For this condition, the input of the mapping function is supposed to be the vectors $\vec{u}$ and $\vec{i}$, thus the mapping function is represented as $f(\vec{u}, \vec{i}) \mapsto \mathbb{R}$.

Aside from user and item, contextual information (time, location, friends' company etc.) is also influential on users' choices. In an RS with concern on the preference variance caused by the contextual factors, the mapping function is represented as $f : \vec{u}, \vec{i}, \vec{c} \mapsto \mathbb{R}$. The additional variable c indicates the context when the iteration happens between user u and item i [4]. The incorporation of context dimension makes $\mathbb{R}$ in the shape of a User Rating Tensor (URT). The context indicator c can be the combination of contextual information like *time of day*, *day of week*, *location* etc. The mapping function makes use of the user profile information, the item meta features and the concrete context condition as the input to give a specific users' preference estimation. Modern RS solutions conduct different assumptions to produce the specific approach to model $f(\vec{u}, \vec{i}, \vec{c}) \mapsto \mathbb{R}$. We introduce the main modeling ideas in RS in the rest of this section.

2.1.1 Collaborative Filtering

Collaborative Filtering (CF) is the most classic approach in RS. The basic assumption of CF is that the users sharing the similar choices in the past can mutually play the role as recommendation reference on the new candidate items. For instance, if A and B bought some of the same books from an online store, when A buys a new book x from the same store, B is supposed with higher chance to buy the same book x [1].

Pure CF deals with URM directly and doesn't need extra information about users or items. It finds every user u the top K neighbors which behave similarly as u. For an item i that u hasn't interacted before, u's K neighbors' interactions with i contribute to the estimated preference of u on i. In this case, the mapping function $f : u, i \mapsto \mathbb{R}$ is represented as the aggregated rating on i from these K neighbored users, i.e. $f(u, i) = \mathrm{aggr}_{(u_k) \in U_k} r_{u_k, i}$ [24]. However, in a system with the scale of millions of users, the time complexity can be $O(MN)$ in the worst case. The sparse representation in the user vector concerning the items makes the similarity calculation less accurate. Thus computational complexity and the sparsity of the user vector are the main bottleneck of user-based CF. Therefore, online retail provider like Amazon advocate apply item-to-item CF in the large-scale system [25]. Item-to-item CF finds the nearest neighbors for items instead of for users. Once the user interacts with a specific item i, RS returns the neighbored items of i as the recommendations. This method increases the speed in real time recommendation and scales independently from the number of users. [21, 26].

However, given the high dimension of users and items, the calculation of the nearest neighbors based on the whole URM is still computationally expensive even for item-to-item CF. Dimension reduction techniques like Singular Value Decomposition (SVD) and Matrix Factorization (MF) are therefore introduced to decompose the URM [27–29]. SVD projects URM onto the latent features space so that users and items are represented by a same group of feature vectors. The direct mathematical decomposition (also called pureSVD) was proposed in recommender systems around the year of 2000 [30]. The direct mathematical decomposition approach treats the missing entries in the matrix as zero, though their actual physical meaning is *unknown* in URM.

As Koren, whose team won the Netflix prize, said in [27], in order to solve the sparsity problem and avoid overfitting (caused by matrix completion), the user-item matrix can be factorized in the shape of an optimization problem. Stochastic gradient descent and alternating least squares are used to resolve this unconstrained nonlinear optimization problem as Eq. 2.1. Here κ means the set of known $user_u$'s rating about $item_i$ in the URM. The dot product of user vector and item vector in the latent factor space directly turns out the final estimation rating (or preference) value. The estimation of $user_u$'s rating on $item_i$ is represented as $\hat{r}_{ui} = \boldsymbol{q_i^T} \boldsymbol{p_u}$, where vector $\boldsymbol{q_i^T}$ stands for $item_i$'s location in the latent features' space, and vector $\boldsymbol{p_u}$ denotes the $user_u$'s location in this space. The regularization term $\lambda(\|\boldsymbol{q_i}\|^2 + \|\boldsymbol{p_u}\|^2)$ is added to overcome the over fitting. The mapping function in RS modeling under MF is then represented as $f(u,i) = \boldsymbol{q_i^T} \boldsymbol{p_u}$.

$$\min_{q^*,p^*} \sum_{(u,i)\in\kappa} (r_{ui} - \boldsymbol{q_i^T} \boldsymbol{p_u})^2 + \lambda(\|\boldsymbol{q_i}\|^2 + \|\boldsymbol{p_u}\|^2) \tag{2.1}$$

Many other factors such as average biases of users and items, implicit feedback involvement, and temporal varying are gradually taken into account for this optimization problem. Extended models like SVD++ [31], timeSVD++ [32] have been proposed. From the comparison by Koren in [27], there is evident improvement on Root Mean Squared Error (RMSE) in Netflix dataset by adding such factors in the extended models during the optimization. However, the optimization target of SVD or MF is to reduce the rating error, the ranking quality or the list-wise performance can't be measured. Thus these approaches don't solve all the problems well in a RS. Authors in [33] proved that item-based K-Nearest Neighbors (KNN) performs better than SVD-based model in recall rate in IPTV's system at the early stage of cold start. KNN outperformed SVD with concerns of involving diversity degree [34]. In addition, SVD model is limited in results explanation, due to the obscure meanings of latent features.

In addition to individual preferences, users are also influenced by contextual factors (like time, location, social suggestions etc.) when they making decisions on items [4]. When only thinking about URM, Koren et al. proposed separating the data set along the time elapsed axis into 30 discrete time bins to distinguish item and user's rating bias temporally. The incorporation of the discrete bias into model-based MF is proved quite effective in improving the prediction accuracy on users' ratings [32]. On top of URM, context information can be added as extra dimensions to form URT. Taking timing factor as an instance, continuous time stamp is categorized into discrete formats (e.g. $T_1 = \{morning, evening\}$ $T_2 = \{workday, weekend\}$), thus the deterministic number of context cases comprise the corresponding dimension entries' indices in the URT. Tensor Factorization (TF) based on URT is a typical way to involve contextual factors in the CF recommender [26,35,36]. In addition, different designs of probabilistic graphical model is also considered as choices for involving contextual factors [37].

Besides users' preferences in the URM, ranking effect is another important prediction target in RS. Learning to Rank (L2R) has been proposed to learn the recommendation model from the ranking of the whole consuming list [6,38,39]. As summarized in [40], L2R methods are divided into three groups: point-wise, pair-wise and list-wise. Point-wise L2R is designed for the labeled ranks, thus multi-classes classification ML models can be used. Pair-wise L2R works well for the recommendation with large amount of candidate items. Owing to the pairs sampling from the lengthy candidates list, time cost can be saved during learning process. List-wise L2R creates the loss function through the cross entropy between the ground truth list and the predicted list. It's more preferable for the shorter candidates list recommendation scenario.

2.1.2 Session-based Approaches

CF based approaches rely URM to produce personalized recommendation results. However, not every online system is able to maintain robust URM or Item Content Matrix (ICM). When users are reluctant to log in to the system, the anonymous interactions makes URM incomplete. Similarly, if less users contribute in generating descriptive content of items in the user generated content based systems, ICM can be quite sparse. These cases happen especially for some online news portals (rare user login) and video platforms (item descriptions missing). For this kind of systems, Session-based Recommender System (SBRS) are an alternative approach to generate recommendations. A thorough survey on SBRS have been given by Shoujin et al. in [41]. According to their categorization, the development of SBRS can be separated into two phases. The first one is the model-free stage from late 1990s to early of 2010s. The second one is the model-based stage, which started since early 2010s until now.

Association Rule (AR) is a typical model-free solution in SBRS. It was earlier proposed in RS to provide recommendations in the non-users identified system [42–45]. AR discovers the association rules on commodities based on the historical transactions. AR discovers rules $X \Rightarrow Y$. If X appears, it's reasonable to deduce the occurrence of Y. X and Y are both subsets of transactions' items set T. Items in all the transactions creates the products set P. The combination of any items in a transaction is denoted as an item set I. I with appearance frequency above threshold *minimum support* (as Eq. 2.2) is a *frequent item set*. Inside each frequent item set, arbitrary pairs of non-overlapping subsets X and Y compose a basic association rule. Every basic association rule owns its conditional probability $P(Y|X)$ for the appearance of Y given the known X. The pairs with conditional probability above threshold *confidence* (as shown in Eq. 2.3) are remained as the final strong association rules $X \Rightarrow Y$. For instance, in an online retail shop, if item set X(Nike shoes, bracer) and item set Y (Spalding basketball) frequently appear together in a same session across all the user sessions, an association rule $(Nikeshoes, bracer) \Rightarrow (Spaldingbasketball)$ could be deduced as a recommendation strategy.

$$support_I = \frac{\#_{I \in transaction}}{\#_{transactions}} \tag{2.2}$$

$$confidence_{X,Y} = \frac{support_{X \cup Y}}{support_X} \tag{2.3}$$

Youtube [1], a famous worldwide VoD website, has applied co-occurrence rate (quite similar with association rules) to generate recommendations for users without identifications [46]. It treats a specific session as a transaction, the videos inside each session as items in AR. For a seed video v_i, its top N related videos constitute its relevant video set R_i. The relatedness score of video v_j to the base video v_i is determined by $r(v_i, v_j) = \frac{c_{i,j}}{norm(v_i,v_j)}$, where $c_{i,j}$ means the co-visited count of $video_i$ and $video_j$ in a same session. $norm(v_i, v_j)$ is the normalization function for the relatedness. The product of the two videos' global popularity $c_i \times c_j$ is the simplest representation of $norm(v_i, v_j)$. A seed set S represents a user's personalized preference, each item i in S has its relevance set R_i. The union of R_i decides the first layer related videos set C_1 ($C_1(S) = \bigcup_{v_i \in S} R_i$). When C_1 is expanded to n layers, the related items set is $C_n(S) = \bigcup_{v_i \in C_{n-1}} R_i$. The final candidate set C_{final} is the union of each layers' candidates sets C_n excluding the original seed set S.

$$P_t(j,k) = p(i_j \to i_k) = \frac{freq(i_j \to i_k)}{\sum_{i_t \in I} freq(i_j \to i_t)} \tag{2.4}$$

$$P(i_1 \to i_2 \to i_3) = P(i_1)P(i_2|i_1)P(i_3|i_2) \tag{2.5}$$

Different from model-free SBRS, which deduces specific rules based on data mining, model-based SBRS relies on specific machine learning models to extract the items' transition relations. Markov Chain (MC) and Neural Network (NN) are frequently researched model-based approaches in RS in the recent years. MC approach models the transitions over items pair within session, and predict the probable next item given a sequence of prior items in a session [47]. The first-order transitional probability from item i_j to i_k is defined as Eq. 2.4. P_t is the $m \times m$ one-step transition probability matrix between m items. With P_0 (the initial probability of all m states) and the transition probability matrix P_t, we can generate the probability of different items in the next time phase. Within a specific session, the chance that user follows a specific interaction path is the joint probability of the first-order conditional probability as defined in Eq.2.5. Aside from the basic first-order MC,

[1]`http://www.youtube.com`

some expansion variants have been also proposed. Latent Markov Embedding approach embed the MC into an Euclidean space and calculate the transition probabilities between items. The transition probability $P(i_{j-1} \to i_j)$ is assumed to be negatively related to the Euclidean distance between item i_j and i_{j-1}. The distance embedding solves the unobserved items pairs in the transition matrix [48].

MC usually tracks the first-order transition relation for pure item pairs in a session, it overlooks the longer dependencies and the context information of a session. Therefore, NN has been proposed to deal with the sequential session-based relations. Shallow NN and deep NN can be both applied to provide recommendation to predict item appearing probability based on session information. Shallow NN encodes user and items session context as input to learn their embeddings on the lower latent feature space with the output prediction layer targeting the next possible item [49]. With such embedding, the items' similarities are calculated based on their representation given specific user and session context. However, shallow NN doesn't track the sequential modeling inside each session. To address this issue, deep NN like normal Recurrent Neural Networks (RNN) and Gated Recurrent Unit (GRU) are applied to learn probability of the appearance of the next-item from the sequential items within sessions [50]. In conventional RNN, based on the state (or session) h_{t-1}, the probability distribution on the items in the next state (or session) h_t is represented as the activation layer as defined in Eq. 2.6, where X_t is the input of the state (or session) t, W are the parameters mapping the input to the activation layer, and U are the parameters reflecting the relation across the sequential neighbored sessions. GRU is a more elaborated version of RNN, it deals with vanishing gradient problem in RNN. For some special items which conduct longer term dependencies session-wise, a forget gate z_t can be learned to decide whether switch to candidate updating $\hat{h}_t$ or keep h_t unchanged as h_{t-1}. The update gate is represented as Eq. 2.8 and the calculation for candidate $\hat{h}_t$ is defined in Eq. 2.9. Inside the candidate calculation function, reset gate r_t (as Eq. 2.10) also needs to be learned [51]. Among many applications of GRU in recommenders, (GRU4REC) is the one gained widespread attention in the industry [52].

$$h_t = g(Wx_t + Uh_{t-1}) \tag{2.6}$$

$$h_t = (1 - z_t)h_{t-1} + z_t\hat{h}_t \tag{2.7}$$

$$z_t = \sigma(W_z x_t + U_z h_{t-1}) \tag{2.8}$$

$$\hat{h}_t = tanh(Wx_t + U(r_t \odot h_{t-1})) \tag{2.9}$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1}) \tag{2.10}$$

As people pay more and more attention on privacy preservation in their online activities, non-identified transactions became the majority in some systems. In such systems, it's uneasy to get the long-term user profiles to provide personalized recommendations. In order to address the issue of missing URM and ICM, session-based approaches are treated with importance in these systems [53].

2.1.3 Selection Bias and Serendipity

Though CF and SBRS provide users with the personalized recommendation strategies. Users' choices and attitudes towards an item can be biased by their own perceptions [54] and lead them easily fall into the *filter bubble*. *Filter bubble* narrows the users' sight to the world they or their similar users could see [55]. Items with higher individual preference are always with higher chance to be recommended. The short heads dominate the recommendation list, while the niche items sleep silently in the long tail. Recommendation models trained on the historical data only reflect the patterns or statistical regularities of users' historical behavior, rather than other potential matching result. Therefore, bringing serendipity into RS to serve users with possibilities on broader choices becomes a hot research area in RS [56–58].

Analyzing information retrieval and recommender system from the perspective of solving a ranking problem, users' biased behavior (like presentation bias, position bias etc.) is mostly modeled based on Learning to Rank approach [10,11,38,39,59]. Usually, propensity weight can be either directly calculated on the shuffled ranking dataset [38] or introduced as model parameter directly into the objective function in Learning to Rank algorithms [39]. The process of eliminating the bias influence is also named as *Counterfactual Reasoning* by Joachims et. al [11]. Though the elimination of bias makes other learned model parameters more representative w.r.t. the observed dataset, the learning objective is still the internal user behavior, thus the lack of fresh event or topic information outside still suffers from the limitation of knowledge.

Some bias modeling approaches involve users' propensity on the rating estimations, particularly in MF and L2R [5,32,60,61]. Inverse Propensity Scoring (IPS) corrects the users' preferences relevant parameters, thus after correction, the model parameters learned reflect users' preference better. Nevertheless, the IPS based bias modeling approaches serve for a more accurate prediction, instead of creating broader reasonable options for users. The causal relationships between novelty, diversity, relevance, timeliness, unexpectedness and users' satisfaction have been confirmed in [58]. Unlike diversity and novelty, which makes recommender compromise to the accuracy metrics, serendipity balances the accuracy and broadness of a recommender. In order to break the selection bias and strengthen the beyond-accuracy factors in the recommendation, approaches like K-Furthest Neighbors (KFN) [62], probabilistic modeling on users' curiosity level [63], semantic web relation between novel and relevant items and user profile [64] have been proposed to counter the selection bias from users. However, it's difficult for these approaches to jump out of the bubble, because they still learn from the internal system recorded data, thus the reasoning clue in the approaches rely on the patterns drawn from the internal systems. Therefore, more external sources involvement is helpful for countering users' selection bias and boosting the serendipity of the recommended content.

2.1.4 Discussion

In this section, we introduced the classical algorithms used in RS. CF, SVD and their variations are applied in recommender systems with user and context distinguishable data. Some ranking algorithms like L2R also base the ranking on the user-context combinations. On the other hand, for the RS without enough user and context information, session-based approaches can model the items sequential appearing probabilities. When analyzing the bias effect for these approaches, no user or context specific bias can be caught.

As to the first recommendation scenario (IPTV system) studied in this dissertation, users and context are both clearly traceable, thus user and context-aware recommendation modeling approach is preferred. To resolve users' selection bias in a recommender, involving the bias factor directly into the modeling process is typical in RS approaches. Nevertheless, these approaches rely on the system internal records, while overlook the serendipity brought by the external factors. Therefore, incorporating external sources into the IPTV recommender will be proposed in the following chapters to properly de-bias the recommendations and bring in serendipity.

2.2 Content Understanding and Popularity Bias

In real e-commercial systems, besides URM, user-profile and item-description are both important descriptive information (in either pure text or categorical meta data form). Content-based RS makes use of such text information to create literally more explainable recommendations. Similarity score calculating approaches from Information Retrieval (IR) domain are also applicable in RS to generate literally closer items [1, 2, 65]. Searching for similar items given user's current interacting item or user's profile in RS is like searching for relevant *documents* given a *query* in IR. Key words or meta data in an active item or in a user's profile are converted as a textual $query$, while other items in the system are treated as documents. The calculation of the similarity between two items or between an item and a user in RS is therefore equivalent to the similarity between a query and a specific document in IR [2, 66].

2.2.1 TF-IDF Similarities

The querying scoring schema involves two important concepts: term frequency and Inverted Document Frequency (IDF). Term frequency denotes the appearance frequency of a specific term in that specific document. If a word's appearance spans a wide range of documents, its representativeness in a specific document is weakened by this wide distribution over all documents, and is penalized by IDF. Eq. 2.11 gives the definition of IDF, where $numDocs$ is the total number of documents in the index, and $docFreq(t)$ stands for the number of documents, in which term t has appeared. The conceptual similarity between a query $\vec{q}$ and a document $\vec{d}$ is defined as Eq. 2.12, where $\vec{q}$ and $\vec{d}$ are the term frequencies representing vectors in the Vector Space Model (VSM). The cosine angle of these two vectors is noted as the similarity. In practice, search engine like Lucene applies the TF-IDF similarity [2] more often as the scoring function. As Eq. 2.13 shows, the final similarity score is the summation of scores from all terms in the query q. For a specific term t, the product of 1) the square root of this term's frequency in the document, 2) the square of the inverted document frequency of the term t, 3) the boosting factor of t and 4) the normalization of the length of the document and q, serves as the contribution of this term t to the final similarity score.

$$IDF(t) = 1 + log(\frac{numDocs}{docFreq(t) + 1}) \tag{2.11}$$

$$cosine(\vec{q}, \vec{d}) = \frac{\vec{q} \cdot \vec{d}}{|\vec{q}| \left|\vec{d}\right|} \tag{2.12}$$

$$score(\vec{q}, \vec{d}) = \sum_{t \in q} (TF_{t \in d}^{1/2} \cdot IDF_t^2 \cdot Boost_t \cdot norm(t, d)) \tag{2.13}$$

2.2.2 Dimension Reduction and Embedding

In the conventional VSM query schema, the key words mapping leads to the missing of semantic understanding of a query process. Semantic representation methods such as synonyms expansion, and hyponyms expansion have been tried to fill the semantic gap [67]. Yet these expansion methods over rely on the pre-defined synonyms and glossaries and the corresponding weighting systems. Latent Semantic Index (LSI) solves the problem by applying SVD on the large term by document matrix. Terms and documents are projected onto the same latent topics' space, where synonyms are located with short distance and polysemy is projected on a more general position instead of one with high offset [68].

In the pure SVD, an $m \times n$ term-document matrix A is decomposed as $A = U\Sigma V^T$, where U is an $m \times m$ unitary matrix, Σ is an $n \times m$ diagonal matrix and V^T is an $n \times n$ unitary matrix. Columns in U are the left singular vectors and rows in V^T are the right

[2] `http://lucene.apache.org/core/8_0_0/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html`

singular vectors. Every diagonal entry in $\Sigma_{i,i}$ is a singular value of matrix A, and is proportional to the importance of the corresponding singular vectors. Usually, only top 10% or top 1% singular values and their corresponding singular vectors are enough to approximate the original matrix, i.e. $A_{m \times n} \approx U_{m \times r} \Sigma_{r \times r} V^T_{r \times n}$. With this dimension reduction technique, the space complexity of a corpus' index is reduced from $m \times n$ to $(m + n) \times r$.

On top of SVD, LSI develops in the direction of probabilistic representation on the latent space. Two typical and widely used text topic modeling approaches are Probabilistic Latent Semantic Index (pLSI) [69] and LDA [70]. In pLSI, there are some basic assumptions: 1) all documents share the same number of topics, 2) the topics inside a document obeys multinomial distribution $p(z|d)$, (z indicates a topic), 3) within each topic, the distribution over glossaries is also multinomial $p(\omega|z)$, (w denotes a term). $p(z|d)$ and $p(\omega|z)$ are the parameters to be estimated in this generative model. Though the whole generation process is complete, pLSI assumes no prior information, which can result in overfitting while learning of $p(z|d)$ and $p(\omega|z)$ from the corpus. To overcome this problem, LDA adds Dirichlet prior to $p(z|d)$ and $p(\omega|z)$ separately. Since Dirichlet and multinomial distributions are conjugate, it simplifies the generative process. The probability density for a multinomial distribution $\vec{\theta}$ given the prior knowledge α is represented in Eq. 2.14, where $\Gamma(n) = (n-1)!$. For document d_m, its topics distribution $\vec{\theta}_m$ is generated by the probability density $p(\vec{\theta}_m|\alpha)$. By the same token, based on prior β, words' distribution $\vec{\varphi}_k$ on a topic k is decided by its probability density $p(\vec{\varphi}_k|\beta)$. With $\vec{\theta}_m$ and $\vec{\varphi}_k$, the generation process goes on the same as in pLSI.

$$p(\vec{\theta}|\alpha) = \frac{\Gamma(\sum_{i=1}^{k} \alpha_i)}{\Pi_{i=1}^{k} \Gamma(\alpha_i)} \theta_1^{\alpha_1 - 1} \cdots \theta_k^{\alpha_k - 1} \tag{2.14}$$

The semantic dimension reduction techniques like SVD and topic modeling are both based on the Bag of Words (BOW) assumption. In BOW, the order of the terms doesn't matter, all the terms are drawn independently and identically. With the introduction of Deep Learning (DL) in the Natural Language Processing (NLP), the embedding and semantic dimension reduction can be realized through the sequential consideration of terms. Continuous Bag of Words (CBOW) and Skip Gram (SG) were the first proposed Word to Vector (W2V) technologies which predict the appearance of an item according to its surrounding context environment. Later in recent research, RNN, Long Short Term Memory (LSTM) and Bidirectional Encoder Representations from Transformers (BERT) have been gradually applied in large text corpus to realize embedding via either single direction or double directions [71, 72].

2.2.3 Popularity Bias and Content Recognition

Aside from individual selection bias which causes the *filter bubble*, *popularity bias* is a more general bias in a recommender. The *popularity bias* reflects a general trending and clustering selections on specific items in a short period of time. The usual understanding of *popularity bias* is based on the item unit. The frequency or the probability of an item is the measurement of the relevant bias processing approaches. For instance, opinion-based CF with weakened weighting functions on popular items has been proposed in [73]. In [74], Abdollahpouri et al. introduced long-tail likelihood added on the probability of an item which is not in the base recommendation list. With this approach, more chances for the niche items are brought in when creating recommendations. Abdollahpouri et al. proved that it's possible to improve the coverage of long tail items without substantial sacrifice of ranking performance in L2R framework by involving dissimilarity matrix into regularization [6]. In addition, fairness inclusive measurements such as disparate impact has also been proposed to overcome the group-level bias [75].

Though the above mentioned approaches are effective for dealing with popularity bias in some scenarios, they all attribute the bias in recommendation to the popularity of specific items. The *granularity* on the item level are discussed as convention when talking about bias in the recommenders. Nevertheless, when looking deeply into the causes of the popularity, it can be hot events, news of an attractive celebrity, specific festivals etc. Thus content level topics extraction and analyzing are also necessary for the understanding of the popularity bias.

2.2.4 Discussion

In some scenarios, the meta-information of users and items contribute as rich textual features for recommender algorithms. Content understanding of these textual meta-information boosts the accuracy of the recommender algorithm, and enhance the human understandable explanation of the recommender. The semantic similarity of the textual items can be generated through either TF-IDF processing or dimension reduction techniques such as *embedding*. In this section, the semantic approaches for content understanding involvement in recommender system are introduced.

More specifically, in recommendation scenario such as digital news portals, the popularity bias presents the temporal pattern since news items are dynamically changing. However, content understanding purely relying on the semantic similarity doesn't contribute to the temporal tracking of the popularity bias. Different from usual semantic similarity representation between items, we focus more on topical bias extraction and ranking tracking to understand the popularity bias in this dissertation.

2.3 Algorithm Recommendation and Evaluation Bias

Though RS was first created and applied in e-commerce system to serve the users with commodities [76], its concept is also applicable to other domains like AS [77] and knowledge graph based co-author inference engine [78]. There are special characteristics different from usual RS applications. More specifically, in AS scenarios, abundant meta-features of problem instances, cold-start condition for the new problem instances are all the characteristics to be taken into consideration when applying RS in AS [79–81]. In this section, we first introduce AS and its application in the recommender system. Thereafter some typical evaluation approaches are listed. Finally, the evaluation and the biased condition in the AS are discussed.

2.3.1 Algorithm Selection and Recommender System

The research problem on how to select an effective or good algorithm given a specific problem instance has been raised since year 1975 by Rice [82]. In this original work, the abstract models of an AS problem have been formulated. Problem space, algorithm space, performance measurements comprise the components of an AS scenario. Some classic solving approach like Single Best (SB), Best Selection for a subclass of problems etc. have been proposed to solve the AS problems. Therewith, AS problem has been expanded to the ranges like computational complexity problems or machine learning tasks [77, 83–85]. With the study and research developed every year, the algorithm space and problem space are continuously growing. To know the exact performance of an algorithm on a problem instance, iteratively executing all the algorithms on the problem instances is usually inevitable. However, such iteration is with very high computational complexity. In this thesis, we mainly focus on the AS scenarios of computational complexity problems like Boolean Satisfiability Problem (SAT), Maximum Satisfiability Problem (MAXSAT), Constraint Satisfaction Problems (CSP), Quantified Boolean Formula (QBF) and Answer Set Programming (ASP) [86–89].

Classic ML methods like classification, regression have proven to be powerful in AS tasks in recent years. Jonhson et al. came up with the domain-independent algorithm composition and selection approach [83]. Through voting on pair-wise performance predictions from different random forest classifiers, $SATZilla^*$ approach has successfully won the first place many times in algorithm selection competitions [90,91]. Instead of only dealing with one classification or regression approach towards multiple algorithms, Lindauer et al. proposed an automatically configured algorithm selector which applies machine learning technique on the pre-processed meta-features of problem instances to automatically generate the proper algorithm portfolio [81, 92]. In some ML scenarios, to automatically select proper algorithms or hyper parameter configuration for algorithm on a specific dataset (problem instance), AutoML tools like AutoWeka [93] and Auto Sklearn [94, 95] becomes popular.

When mapping the problem instances and algorithms to users and items in RS, typical approaches in RS are also applicable in AS scenarios. Since 2010, Stern et al. used Bayesian Matrix Factorization (BMF) (originally designed for RS) in AS scenarios and got some good results concerning the algorithm selection effect [96, 97]. Thereafter, many researchers attempted the approaches from RS to solve AS tasks. Misir and Sebag created Alors AS system, which utilized random forest to map meta-features of problem instances onto the latent feature space. Based on these latent features, their CF is designed to make algorithm recommendation [77, 98]. Yang et al. proposed Principle Component Analysis (PCA) to decompose the performance matrix actively to solve the sparse performance entries problem for the new problem instances [99].

Since AS still faces the issue of solving the ranking of algorithms, Learning to Rank (L2R) approaches [6,38,39] are also suitable to be applied in AS scenarios. As summarized in [40], L2R methods are divided into three groups: point-wise, pair-wise and list-wise. Compared with point-wise and pair-wise approaches. List-wise L2R is more preferable for the shorter candidates list, thus is chosen to solve the ranking issues in the AS problems. The attempt of combining CF and L2R in AS problems was initiated from [77, 97, 100]. Through segmenting the values of entries into L intervals, ordinal regression can be applied to learn the segmented rank. In [100], under the L2R framework, sigmoid function plays the role of a pair-wise surrogate which embeds the polynomial scoring model function, and the model is also proven to increase the algorithm selection quality from algorithms' pair-wise performance ranking perspective.

2.3.2 Evaluation in Recommender System

There are multiple measurements which evaluates the effectiveness of an RS. These quantified metrics reflect the Quality of Experience (QoE) from different perspectives. Depends on the way to understand users' preference, we separate these metrics into three classes: *rating and error*, *ranking and precision* and *user-centric evaluation* [101]. These evaluation metrics in RS also guide the evaluation metrics in AS.

Rating and Error In RS, error metrics are used to assess the accuracy of rating estimations on the unknown items in the URM. Mean Absolute Error (MAE) and RMSE are the metrics frequently used in the experimental evaluation phases.

- **Mean Absolute Error** calculates the mean absolute error between estimated rating value and real rating value in the test set. As Equation 2.15 shows, in the test set, for every pair between user u and item i $(u, i) \in TestSet$, the absolute error between real rating $r_{u,i}$ and the $\hat{r}_{u,i}$ are summed up and is then averaged by the number of such pairs in the test set [102].

$$MAE = \frac{\sum_{(u,i)\in TestSet} |r_{u,i} - \hat{r}_{u,i}|}{|TestSet|} \tag{2.15}$$

- **Root Mean Square Error** As RMSE is the final evaluation metric in the Netflix prize in 2009, many researchers are driven to improve algorithms to adapt to this error metric [27, 32]. As Eq. 2.16 shows, RMSE is defined as the root of mean square errors of each pair of estimated $\hat{r}_{u,i}$ and real $r_{u,i}$.

$$RMSE = \sqrt{\frac{\sum_{(u,i)\in TestSet} (r_{u,i} - \hat{r}_{u,i})^2}{|TestSet|}} \tag{2.16}$$

When applying error metrics in the recommender evaluation, the errors are modeled as the loss objective in the optimization algorithms. However, precisely predicting users' rating doesn't directly reflect the QoE or users' satisfaction.

Ranking and TopK Shortly after RMSE has been broadly researched in the RS, Cremonesi et al. emphasized the importance of measuring the recommender from the perspective of the top K elements in the recommendation list [103]. Classic evaluation metrics in IR are also adaptable for clicks of top K elements assessing in recommenders. Precision rate [104, 105] and recall rate [22, 33] are most prevalent and normal metrics introduced from IR to RS. Furthermore, Mean Average Precision (MAP) and nomarlized Discounted Cumulative Gain (nDCG) which take position weights into account are also important evaluation metrics [106, 107]. In the real system, only top K recommended items are shown to the user, these ranking based metrics are calculated specifically for these top K positions.

- **Precision Rate** denotes the hit ratio of relevant items in a recommendation list. Relevant items are the ones which have been clicked or viewed by users. As shown in Eq. 2.17, it's the Precision Rate (PR) on a single transaction, where $\{RecommendedItems\}$ represents the recommended list. The ratio of the intersection of $\{RelevantItems\}$ and $\{RecommendedItems\}$ w.r.t. the recommended items is defined as the precision. If top K is limited, the number of set $\{RecommendedItems\}$ is K.

$$PrecisionRate = \frac{|\{RecommendedItems\} \cap \{RelevantItems\}|}{|\{RecommendedItems\}|} \tag{2.17}$$

- **Recall Rate** is defined as Eq. 2.18. The intersection between recommended items set and relevant set denotes the hit items. The set $\{RelevantItems\}$ contains all relevant items a user interacted in test set. Recall Rate (RR) is also useful to judge the algorithm's effect on waking up new items in cold start period [22].

$$RecallRate = \frac{|\{RecommendedItems\} \cap \{RelevantItems\}|}{|\{RelevantItems\}|} \tag{2.18}$$

Precision and recall are the metrics calculated based on set operations. Nevertheless, the position of the relevant items and in the recommended list also matters to the users. When the relevant items are with higher rank in the recommended list, it hints the better recommendation quality than the condition that relevant items are captured but with tailed rank. To distinguish the ranking quality with positions, MAP and nDCG have been also applied in the recommender evaluation.

- **Mean Average Precision** is a ranking version of PR. It's influenced by the inner rank of the result list. The calculation of $MAP(K)$ is divided into two steps. The first step is to acquire Average Precision (AP). As Eq. 2.19 shows, $AP(K)$ is the average precision value from $P@1$ to $P@K$. If the k_{th} item in the recommended list is not a hit, $P@k$ is excluded by multiplier $rel(k)$, an indicator value for the hit item. The average precision of all the relevant hit items forms the $AP(K)$. AP is sensitive to the ranking position of the relevant items. For instance, facing two 4-items result lists $\{0, 0, 1, 1\}$ and $\{1, 1, 0, 0\}$. The precision rate $P@4$ of these two list are both 2/4=0.5. Yet when calculating AP, value of $AP(4)$ of the first list is $(0 + 0 + 1/3 + 2/4)/2 = 5/12$ while the $AP(4)$ of the second list is $(1/1 + 2/2 + 0 + 0)/2 = 1$. Obviously, AP tells the recommendation quality with the consideration of the ranking positions.

$$AveP(K) = \frac{\sum_{k=1}^{K}(P(k) \times rel(k))}{|\{Relevant \quad Items \quad in \quad Recommended \quad Set\}|} \tag{2.19}$$

However, $AP(K)$ is calculated for one specific user. For a general performance of the whole system, the mean value of $AP(K)$ for all users is the final measurement. As defined in Eq. 2.20, $MAP(K)$ reflects the whole system's precision rate and ranking quality.

$$MAP(K) = \frac{\sum_{u=1}^{U} AveP_u(K)}{|\{U\}|} \tag{2.20}$$

- **Normalized Discounted Cumulative Gain** The basic idea of nDCG is originated from Cumulative Gain (CG): $CG_K = \sum_{i=1}^{K} rel_i$, where rel_i indicates whether $item_i$ is relevant (1) or not (0). CG_K is the total count of relevant items in the top K recommended set. On top of CG_K, DCG_k added the penalty factor to the lower ranked relevant item and is denoted as $DCG_k = \sum_{i=1}^{K} \frac{2^{rel_i}-1}{log_2(i+1)}$. $\frac{1}{log_2(i+1)}$ is the penalty factor to reduce the influence of lower ranked hit $item_i$, it makes DCG more sensitive to the ranking positions of the relevant items. To normalize the DCG_K, ideal DCG_K ($IDCG_K$) is proposed. $IDCG_K$ supposes the most ideal condition that every item in the top K recommended list is relevant (i.e. each entry rel_i is 1). When the real DCG_K is divided by the ideal one $IDCG_K$, the calculation of nDCG is represented as $NDCG = \frac{DCG_K}{IDCG_K}$.

System Utilities Evaluation Though top K ranking based metrics evaluate the list-wise recommendation quality, it doesn't tell directly the benefits created by recommenders for the whole system. In addition, users' subjective opinion can't be reflected from these quantified values. Thus other system-wide benefit evaluation metric has been broadly accepted in the real commercial RS. Some special user-involved metrics like CTR and browsing time on the menu could reflect the system-wide cost due to users' patience.

- **Click Through Rate** utilizes user's click behavior to measure how successful the recommended result is. For a specific item in a system, the fraction of its clicked times out of its recommended times reflects how users are satisfied with the items recommendations. In its definition $CTR = \frac{Clicks}{Impressions} \times 100\%$, $Impressions$ stands for the times a specific item shown (recommended) to the users, while $Clicks$ denotes users' clicks after seeing the recommendations. Since CTR reflects the system wide benefit, it is adopted in many commercial systems as the interest evaluation metric. Youtube [46], LinkedIn [3] [107] and Plista [4] [108] use CTR as the main evaluation approach.

- **EPG Browsing Time** tracks how long it takes for a user to find his or her preferred program, particularly in IPTV system. The longer it takes, the lower the *QoE* is. In [109], Song et al. conduct experiments on the expected EPG Browsing Time (EBT) and prove that RS contributes to a better QoE in personalized IPTV. For linear TV broadcasting without recommendations, if there are N channels in the system, the probability of a user finding his or her preference at first time is $\frac{1}{N}$, and the probability of finding it at the second time (after once switching) is $\frac{N-1}{N} \times \frac{1}{N-1} = \frac{1}{N}$. Every time a user switches a channel in a linear TV, there is only $\frac{1}{N}$ chance to meet the preferred one. If the duration of switching a channel and judging willingness is noted as t', then the theoretical Electronic Program Guide (EPG) browsing time after switching i channels is $(i-1)t'$. The expected EBT is $E_{norm}(EBT) = \sum_{i=1}^{N} \frac{1}{N}(i-1)t'$. With the help of RS, the opportunity

[3]https://www.linkedin.com
[4]https://www.plista.com/

that a user meets the preferred program is proportional to precision P_a of the RS rather than the uniform distribution $\frac{1}{N}$. The probability that first $i-1$ results are not relevant but the i_{th} item is a hit is $(1-P_a)^{(i-1)}P_a$. The total expected EBT is $E_{rec}(EBT) = \sum_{i=1}^{N}(1-P_a)^{(i-1)}P_a(i-1)t'$. With higher P_a, the expected EBT $E_{rec}(EBT)$ is much shorter than the expected EBT under the linear browsing time. The saved time spent on the menu browsing creates the increased QoE in IPTV system.

2.3.3 Evaluation Bias in Algorithm Recommendation

In this thesis, we mainly deal with the AS problem in computational complexity scenarios like SAT, MAXSAT, CSP, QBF and ASP [86–89]. In these AS scenarios, *run time* is the performance indicator for all candidate algorithms. The *prediction error* on the performance like *runtime* and *solved ratio* of the predicted optimal algorithm are the main effect measurements for AS approaches [80, 110–112]. Therefore, the error metrics, accuracy metrics, ranking metrics mentioned in the Subsection 2.3.2 are also applicable for the AS problems.

Though accuracy and error oriented evaluation targets are quite important in AS, other evaluation factors like inference time also presents its power to tell the quality of an AS approach. For instance, some benchmark AS approaches create algorithm dependent models, which leads to much longer inference time than the single model based AS approach. If only accuracy factors are involved in the evaluation protocol, the recommendation results will be biased towards the accuracy, yet other metrics are overlooked. Therefore, a balance between the accuracy relevant metric and other evaluation factors is expected to counter the evaluation bias. In addition, for conventional AS evaluation protocols, usually only the $TOP1$ algorithm in the predicted list is selected and compared with ground truth. This $TOP1$ selection of the candidate set causes the evaluation results being biased by a single selected candidate. Applying the evaluation on the $TOPK$ candidates set is helpful to overcome the potential noisy and bias on the exact $TOP1$ protocol.

2.3.4 Discussion

In this section, we focus on the research background of algorithm selection, a recommendation scenario which is not user oriented. Given the similar data input and output, a typical AS problem can be mapped to the cold-start phase in a RS problem. Thus classic RS algorithms like CF, MF and L2R can be applied to solve an AS problem. Therefore, the traditional accuracy-oriented evaluation metrics in RS are also applicable in AS.

However, in AS scenarios, purely relying accuracy-oriented evaluation metrics may result in the evaluation bias. Aside from accuracy, other factors like reducing the inference time also benefits the algorithm selection process. To counter the evaluation bias in AS problems, we will introduce multi-objective evaluation metrics and $TOPK$ expansion in the following up chapters in this thesis.

2.4 Conclusion

In this chapter, we introduce the background information and literature review for the RS. Both state-of-the-art related work and the relevant bias issues have been discussed. First, we summarize the recent literature overview on the user, item and context concepts in RS. The selection bias and serendipity recognition have been also discussed. Second, we explain the ways of content and semantic understanding in the AS, and the correlation of such understanding with the popularity bias. In the end, we researched the recent work on solving AS problems under the RS framework. The evaluation bias issues in this domain has also been listed.

The following chapters are organized in the same order as the structure in this chapter. Solving the contextual selection bias in TV program recommendation is researched in Chapter 3. Topic level understanding of popularity bias in the news recommendation is represented in Chapter 4. Finally, Chapter 5 focuses on the algorithm recommendation and the balancing of evaluation bias.

CHAPTER 3

IPTV SYSTEM: MODELING AND COUNTERING CONTEXTUAL BIAS

Recommender System (RS) makes use of information from users, items, and context to build up a scoring model to help users find what they could like in different occasions. Contextual influence on the effect of recommender systems has been broadly researched in recent years [15, 26, 113, 114]. Typically, in an IPTV system, users' preference on the video content alters along with the contextual factors (time, location, company by other people etc.) switching. However, the context-aware recommendation algorithms proposed by now only assume the contextual influence as a global fixed value in the model, yet users' individual subjective opinion on different contextual factors and their biased choices based on the contextual factors are always ignored. In this chapter, we first illustrate the difference in users' individual attitudes due to the contextual factors' influence by conducting a user study of IPTV users. Based on this proved assumption, we build up our own context-aware recommender model which can involve users' personal inclination on contextual factors into the recommendation model. Though the proposed model performs better to fit the user behavior data, given specific *contextual factors*, the strong biased choices observed in the system makes the model unavoidably falls in the *filtering bubble*. To counter this bias which leading the fitted model into unbalanced, we propose introduce the external media resource like Twitter in the same domain to play the role as another recommendation source.

3.1 Introduction

The increasing availability of broadband Internet and consumer electronics such as Smart TVs transforms how audio-visual content is consumed. Though User Generated Content (UGC) attracts more attention nowadays among the younger generations, scheduled linear TV is still the main source for online TV content, and maintains a significant market share [115, 116]. In order to enhance users' stickiness to the IPTV system, recommenders are favorable to be embedded in online TV platforms to provide users with potentially interesting items [117].

According to the survey conducted for TV viewers' behaviors [118], more than half of the relevant determinants for program selection depend on the *situational context* of the user. Furthermore, 60% of respondents state that the presence of company and the available time are important contextual factors to select a program. Context-aware Recommender System (CARS) algorithms, e.g. tensor factorization [26, 113] and Probabilistic Graphical Model (PGM) [15], incorporate time [36, 37, 114] or location [119, 120] as additional parameters to encode contextual information. In addition, some advocate to model the local social environment [121, 122] and propose strategies to improve recommendations in households with multiple users sharing a single device [123].

In this chapter, we investigate the contextual influence in the TV recommendation strategies and the corresponding contextual bias treatment by solving the following Research Problems (RPs).

RP3.1: What is the role of contextual factor in users choices in IPTV services? Though many modern research directly build up recommendation model with consideration of contextual information from the historical user interaction data [26, 37], there are rare study about users' own opinion on those contextual factors on the recommendation effect. To make up for this gap, we conduct a user study to understand the role of contextual factors in users' eyes, thus find the clue for recommender design. The questionnaire feedback and focus group discussion result of this user study shed some light on the individual difference in the inclination on contextual factor. Meanwhile, social influence and breaking events are also mentioned as important factors.

RP3.2: How do we model different contextual factors in one model while catching the corresponding users' inclination on the contextual factors? Though contextual factors received the much attention and have been involved in many recommendation models [26, 113, 114], the contextual factor in the model are usually modeled fixed instead of user specifically [32,34]. In this chapter, we extend the LDA to a context-aware version to create the recommendation model. With the help of the added probabilistic selection node in the original probabilistic graph, this extended LDA model can also capture users' inclination on different contextual factors. Thus the context-awareness is modeled individually different. The experimental results proved that this personal inclination involvement can balance the accuracy and diversity when concerning the recommendation effect.

RP3.3: How to counter the contextual bias in TV recommendation scenario? When thinking outside recommendation model, users' *biased choices* make the machine learning algorithms prone to recommend the already skewed information to reach higher accuracy, and causes the *filtering bubble* issue [124]. In TV domain, due to the programs airing schedule, such bias is more reflected in the contextual form. Though bias has been taken into account in both recommendation model and the recommendation user-interfaces design [5,6,125] recently, the focus on the internal system user data still unavoidable limit the recommender's broadness. Therefore, we propose extend the notion of context to domain relevant trends detected in online social media streams like Twitter, and make use of this external social influence to broaden the users' choices. The peak and trend information of specific program is extracted as the indicator to do the recommendations. The experiment shows that introducing external social influence can alleviate the *contextual bias* without violating users' behavioral habit.

To sum up, in the process of solving the above mentioned RPs, our contributions in this chapter include: 1) a user study help uncover the role of contextual factor from user's sight; 2) an extended probabilistic model which can incorporate users' inclination on contextual factors and balance the accuracy and diversity of the recommendation result; 3) introducing the relevant social media stream as the external contextual factor and making use of it to counter contextual bias caused in users' TV watching behavior.

The sections in this chapter are structured as following: a general introduction is given in Section 3.1, after which we conduct a user survey to confirm the existence of the contextual information influence from users' view in Section 3.2. Following that, Section 3.3 models the context in user watching behavior as a PGM, which extends the LDA. Though the extended probabilistic model performs well on both accuracy and diversity metrics, the contextual biased choices still exist in the dataset. Thus approach incorporating programs related trend from heterogeneous social system is introduced in Section 3.4. Lastly, conclusion is given in Section 3.5.

3.2 Users' Understanding Towards Context Influence

In this section, we investigate the role of contextual factors from the individual users subjective intentions in IPTV systems. The existing user studies on IPTV user behaviors usually focus on social network involvement, users' churning behaviors, users' satisfaction factors and selection criterias [126–129], yet the influence of contextual factors are seldomly discussed in the published user surveys. Therefore, we design our own user study, an online questionnaire via Google form to gather IPTV users' feedback towards the context influence. The questions designed in our survey mainly target the topics: 1) users' general preferences on TV genres altering on the temporal factors, 2) users' inclination on the limited recommendation types (user habitual watching content, breaking events influence and social influence) change along with context regarding time and location, 3) the correlation between different contextual factors according to the user's choice on the importance. The analysis of the online questionnaire feedback and the following up focus group discussion confirmed the facts that users' preference on TV content is biased in different contextual factors.

Table 3.1: Demographic information of questionnaire respondents.

Question	Options	N	%
Gender	Female	15	29.4
	Male	35	68.6
	N/A	1	2.0
Age	15~20	1	2.0
	20~30	38	74.5
	30~40	12	23.5
Profession	Employee	19	37.3
	Freelancer	2	3.9
	Teacher	2	3.9
	Student	26	51.0
	other	2	3.9
Place of Birth	Europe	17	33.3
	Asia	33	64.7
	N/A	1	2.0

The online questionnaire remained valid throughout the month of August 2013, with a total of 51 respondents in the end. The demographic information of the questionnaire respondents is listed in Table 3.1. All respondents are digital natives, i.e. were born after the start of the digital age (around 1960); so their understanding of legacy TV (terrestrial, cable, satellite) and IP-based TV services was clear. Most participants are either employees or students. In terms of the place of birth and residence, Asians and Europeans form the two largest groups of our respondents. This coincides with the survey result from Point Topic [130], which shows that Asia and Europe are the two biggest markets for IP-based TV content with 48.7% and 36.6% market share of the worldwide IPTV subscribers, respectively.

Table 3.2 represents TV usage habits of the respondents. We observe that a large majority consumes IP-based TV services much more than traditional TV, with more than half of the participants spending at least five times more time on IP-based TV services than on normal TV. Moreover, 86.3% (19.6+25.5+41.2) report that they have been using IP-based TV services for over two years. These statistics confirm that the respondents to the questionnaire represent experienced IP-based TV service users, possessing the required reference value for our survey.

Table 3.2: Respondents' basic usage information.

Question	Options	N	%
Watching Duration Proportion (Normal TV: IP-based TV service)	5+:1	6	11.8
	4:1	3	5.9
	2:1	1	2.0
	1:1	3	5.9
	1:2	4	7.8
	1:4	8	15.7
	1:5+	26	51.0
How many hours per day do you use IP-based TV services?	0-1 hours	30	58.8
	1-3 hours	17	33.3
	3-5 hours	4	7.8
How long have you been using IP-based TV services?	Never	1	2.0
	< 6 months	4	7.8
	1 year	2	3.9
	2-3 years	10	19.6
	3-5 years	13	25.5
	> 5 years	21	41.2
Prefered IP-based TV services	Set-top box IPTV	15	29.4
	WebTV / Internet TV	43	84.3
	TV Apps or softwares	16	31.4

3.2.1 Users' Selection Bias on Genres Towards Contextual Factors

The first question we designed tries to capture the changes in users' interests on TV programs given different temporal contextual factors: *"What kind of programs would you prefer watching a) in the morning, b) during a break at daily work, c) in the evening and d) on weekends?"* As presented in Figure 3.1, twelve basic genres of TV content were listed as choices specifically for the four categorical time periods. During each timing period, users may select their preferred categories as many as they want. Thus for every TV category in each timing period, at most all 51 respondents would vote for it.

In line with the intuitive reasoning, we observe the following trends despite of their resemblances to most TV providers strategies: i) weather report and daily news seem to be favorable choices in the morning or during a break at work, when people usually spend much less time watching TV; ii) similarly, during a break at work, those relatively short TV content such as daily news, sport, music and entertainment content are often consumed; iii) users' preference in the evening and on weekends show very similar pattern: comparatively

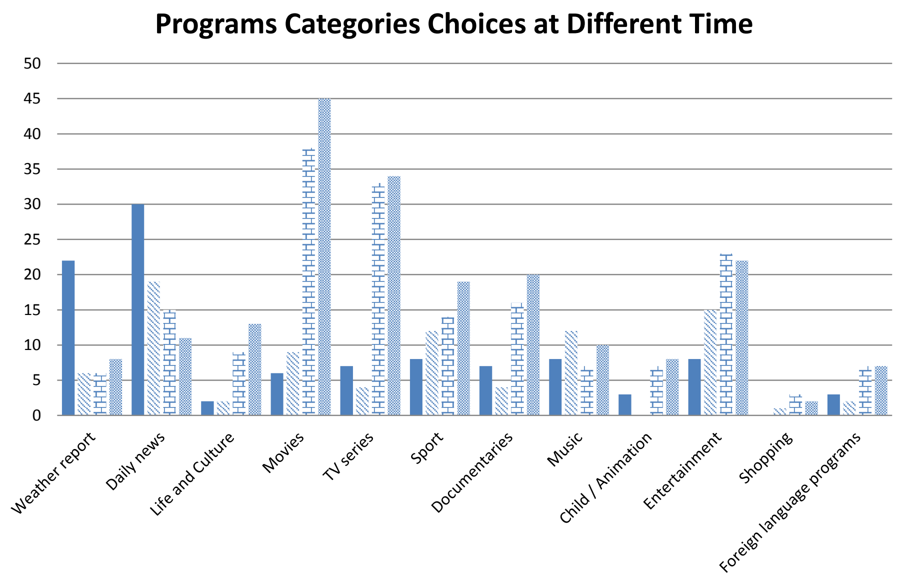


Figure 3.1: Users' choices on program categories at different time of day.

longer programs such as movies, TV series and documentaries outweigh other content types.

The next question that we cover is on the user's direct opinion on a more limited set of recommendation types given to them in a set of changing contexts combination regarding time and location: *"Consider three types of content recommendation types provided to you at the same time (habitual content at this time, breaking news or events happening just now, friends' instant suggestions). In each of the contexts (at home in the morning; at work hours during a break; at home in the evening; on weekends), which of those recommended types are you most likely to choose for watching?"* As depicted in Figure 3.2, the users are much more interested in hearing about breaking news and events during work hours or in the morning at home. Conversely, the habitual content or friends' suggestions becomes much more favorable in the evening or on weekends. In other words, the influence of contextual factors on users' choices on recommendation types like breaking news, social company and users' habitual content may change with alterations in certain contexts such as time and location. It illustrates that users' inclinations on contextual factors presents individual differences, meanwhile are also dynamically changing.

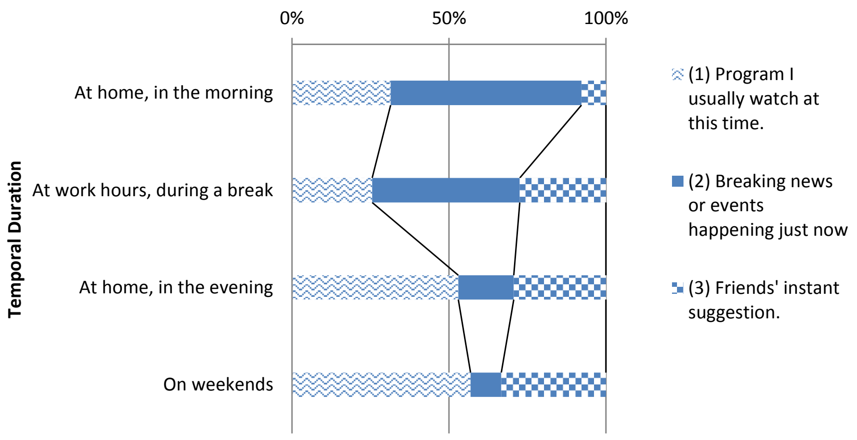


Figure 3.2: Contextual factors' influence on recommendation types.

3.2.2 Importance of Contextual Factors in Users' Eyes

Targeting on the four proposed contextual factors ("*Current Time*", "*Current Location*", "*Social Accompany*" and "*External Breaking Events*"), added with user's "*Daily Viewing Habit*", we provided five factors in the question "*How important is each of the following factors regarding their influence on your own (subjective) choice of TV programs recommended?*". The importance for each of them ranges as "*1-Not Important, 2-Somewhat Important, 3-Can't decide it yet, 4-Important, 5-Very important*". In Figure 3.3, the distribution of respondents' ratings on the importance of these factors are shown. Interestingly, different from what we expected, there is no significant centralized trend of users ratings. For each factor, the number of respondents who rate positively (more than point 3) doesn't outweigh the number of respondents who rate negatively (lower than point 3) that much. And there indeed exists a certain number of respondents who would like to rate as "3-Can't decide it yet". The approximately middle (point 3) average score for each factors' importance ("*Daily Viewing Habit*": 2.78, "Timing": 2.98, "Location": 3.39, "Social Accompany": 3.25 and "External Breaking Event": 3.29) further illustrated that users have no certain unified alike opinions towards the importance of contextual factors. Where there are users incline to a factor, there are users reject to that factor. Therefore, contextual factors' influences are individually different, there is no specific contextual factor that really matters to everyone.

More in detail, we calculated Pearson Correlation Coefficient for each pair of factors by making use of respondents' rating scores on each factor. Figure 3.4 shows that correlation between every two factors are all quite weak, even the highest Correlation Coefficient is only 0.48 (turned out between "*Social Accompany*" and "*External Breaking Events*"). This indicates that for users, these five factors have different meanings thereby they should be treated separately as independent factors.

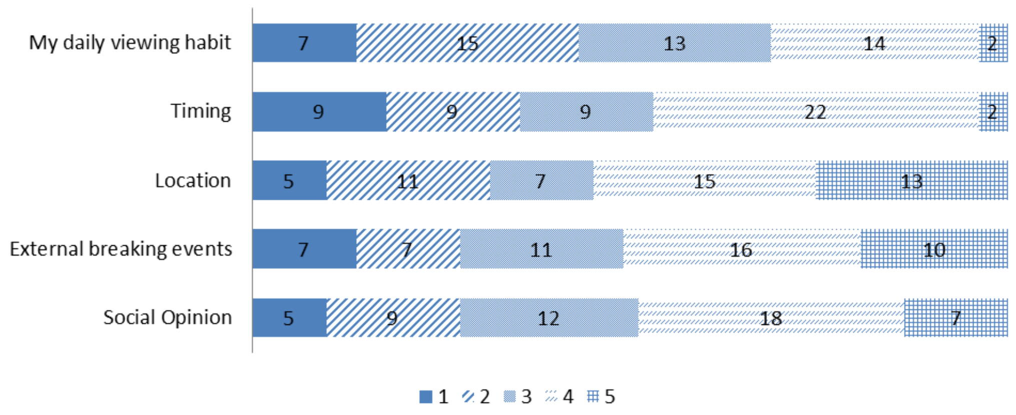


Figure 3.3: Users' scoring on factors' importance.

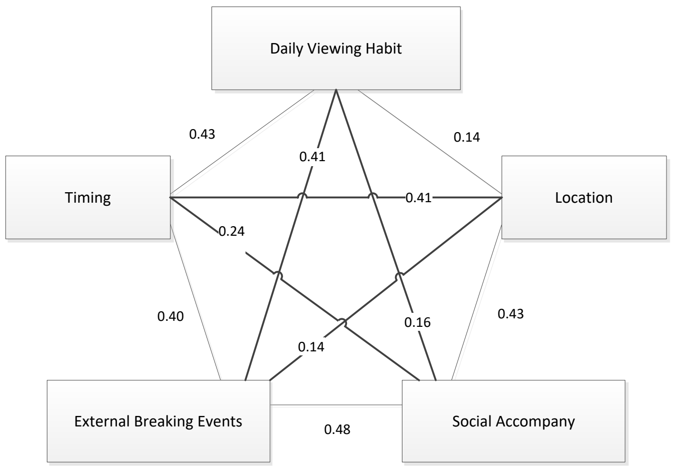


Figure 3.4: Pearson Correlation Coefficient of different pair of factors importance.

In short, since all these five factors are influential to users at different degrees, their effects on recommenders should be considered personalized. At the same time, the weak correlations between the pair-wise factors indicate that they need to be treated individually.

3.2.3 Users' Subjective Proposal on Contextual Factors

To compensate the implicit options in the questionnaire, we raised two extra *subjective* questions - **Q1**: "*From your point of view, what features should a perfect IP-based TV recommender offer?*", and **Q2**: "*What other contextual factors may influence your decision to follow a certain program on IP-based TV services?*". Contrary to what we expected before releasing the questionnaire, many respondents took part in these non-mandatory questions. Summarized in Table 3.3, these subjective feedbacks are not scattered as pieces but are showing some central tendency. We signed the total number of each question's responses as "N", and counted numbers of each main opinion targeting the specific question as "F(N)" to do the statistics.

Table 3.3: Feedbacks of subjective questions.

Question	N	Feedback Content	F(N)
(1)From your point of view, what features should a perfect IP-based TV recommender offer?	22	*Programs on breaking or hot events should be reminded.*	6
		Social interaction with friends.	6
		Knowing my interests and habits.	5
		Other features as program quality and variety, less advertisement, free of charge, clear and fast interface.	6
(2)What other contextual factors may influence your decision to follow a certain program on IP-based TV services?	16	*Meta data of a program content (theme, guests, popularity, preview, entertainment and so on.)*	5
		Whether the length of a program is suitable for my current status.	2
		My mood while watching programs.	2
		Program content in popped windows or advertisements happened to be seen.	2
		Factors which we already proposed (as temporal regularity, social opinions, hot news influences).	6

From feedbacks of **Q1**, we see that our proposed contextual factor of "*External breaking events*" and "*Social interaction*" coincidentally match users' subjective requirements for a perfect IP-based TV recommender well. In addition, program's variety, less advertisement, free of charge, clear and fast interface are also respondents' subjective suggestions on the important contextual factors.

Responses of **Q2** present a better understanding of proposed influential contextual factors. Interestingly, many respondents wrote about the quality of TV programs. As one respondent replied, "*From my point of view, players, guests, theme and recreational are features which can determine my choices*". Another respondent also commented, "*I care most about the nutrition of the TV content and don't want to be bothered by some content with low quality but high popularity*". Though these are attributes of TV contents but not contextual factors, it illustrates that users have begun to evaluate programs from a semantic or meta information perspective, which is highly advanced and quite selective. Interestingly, two respondents mentioned that factor of "busy or not" decides their preferred length of TV programs. One of them wrote, "*Busy or not is an important factor, when I'm free, I would like to watch long programs. On the contrary, short program segment will be preferred if I'm busy*". It is quite inspiring to know the possible relationship between "busy status" and "program's length". Moreover, mood is another innovative opinion suggested by respondents, "*My mood will influence my choices, e.g. when I just broke up with my boyfriend, only sorrow music could bring me consolation*". As an invisible factor, mood is not so easily to be caught, but if any clues can reflect users' mood status, that will be valuable as a reference for recommendation. Besides, suitable UI as popping-window or advertisement frame is also elements that can arouse users' attractions.

3.2.4 Focus Group Discussion

Questionnaire statistic demonstrates a general condition of users' inclination, while if we would like to know how they think on earth, sitting together and face to face discussion is the most effective approach. The reason for users' high/low rating on specific factor, their supplement to our missing points or other potential contextual factors, all these feedbacks are what we expected from this focus group discussion. Hence we delivered the invitation to respondents of the questionnaire, and finally 8 of them made it to participate in this discussion. In the afternoon on 11th October 2013, the focus group discussion was organized. In this section, we number the 8 participants from 1 to 8 as P1 to P8 to retell their valuable ideas.

Timing and location determine busy or not - Even though timing and location are two hot contextual factors which are studied in many works and many respondents signed them as highly relevant contextual factor which influence their watching behavior, our participants gave quite innovative opinions on these two factors. Participant 3 (P3) started by negating the influence of timing factor while being quite supportive to location factor, "*If I decide to watch TV content from internet, it means I'm idle and that can only happen at home. Whenever I'm in the office, I'll be busy with my work and I'll not open any TV related applications. Since my working hour is not that regular, exact timing can not help tell whether I'm busy, but location perhaps can more or less give some information*". Afterwards, P1 also expressed his view, "*From my opinion, whether I'm working or on vacation will result differently on my willingness to accept recommendations. But for me, time of the day, day of week can regularly determine my status of busy or not*". To this point, P6 also has his own experience, "*Whenever it is, wherever I am, only if I choose to use IP-based TV services to have recreation, it means I have time and will enjoy the content I'm interested in*". From the statements of these three participants, an interesting common feature can be extracted from their sayings: busy or not, that is the most concerned issue when talking about timing and location. In other words, busy or not may be the direct decisive factor when users watching TV content, while timing and location are factors which can be measured in real environment to deduce users' status of busy or not and thereby give the prediction. This conclusion is quite similar with some users' subjective opinions in the questionnaire, which makes it more reference valuable.

Comments on social accompany - When the discussion topic moved on to "*social accompany*", participants provided insightful opinions again. P8 and P2 both showed high interest in social interactions during IP-based TV services usage, "*It will be a great experience if I could exchange ideas with my friends about our common interested programs, and I always get proper suggestions from my friends on TV content*", said P8. And P2 supplemented, "*Yes, especially during soccer games, I enjoy the experience of chatting with friends and share our opinions*". Whereas on the totally opposite side, P3 really can't find reasonability of social accompany when watching TV content, "*When I watch TV content, I really hate being bothered by others, I just would like to be immersed in my own interested content silently*", he stated. This saying aroused P8's complementary to his former statement, "*I also would not like to be interrupted when really watching something, but I enjoy the social interaction only for those interesting content, from which I need fun and laugh*". Then P1 also joined in this talk, "*For me, both TV content directly suggested from friends and their own interested content will be attractive. To know what my friends currently are watching is also quite interesting in my case*". From these sayings, we are reminded that preference on social accompany is not only individually different, but also will be dynamic for a specific user (as P8). Therefore, occasions and content categories should be carefully considered when deciding to provide social-based recommendation.

Users' sensitivity to content popularity and quality trade-off - Opinions on popularity and quality of TV content, as appeared in subjective responses of the questionnaire, were incidentally proposed again in the focus group. *P4* stated, “Popularity is a useful reference when I choose TV content, while it won’t work sometimes since content’s popularity can’t directly determine its quality." *P2* continued, “I also found that some so-called popular TV content are pushed in front of us only due to commercial reasons rather than users’ preference." Apparently, users are more sensitive to TV content’s quality now than ever, and they won’t be satisfied with just the popularity statistics. Thus distinguishing high quality content would be quite an important aspect for recommenders.

3.2.5 Discussion

In this section, we discover users’ opinions about contextual factors involvement when recommending TV content through IP-based TV services. To this end, we designed the questionnaires based on four contextual factors and users’ habitual preference. The result turns out that users’ attitudes towards contextual factors’ influence are individually different and dynamically changing. Referencing these results and the focus group discussion, we answer the question **RP3.1: What is the role of contextual factor in users choices in IPTV services?** as following: 1) users preference on TV content and recommendation types varies while context changes, specifically varies individually; 2) timing and location are the important contextual factors; 3) social influence, breaking news and events deserve our attention while making recommendation strategy. Guided by the insight drawn from this section, the context-aware model built in the following section should meet the requirement of being capable of capturing the individual difference of the contextual factor inclination.

3.3 Modeling Contextual Factors in TV Program Recommender

Inspired by the result of the user study in the last section, we know that users’ inclinations on contextual factors are individually different and dynamically changing. Thus we need a context-aware recommendation model which has the capability of capturing such individually altering inclinations. Therefore, we build a context-aware Probabilistic Graphical Model (PGM) which can integrate all candidate contextual factors into one model and include users’ individual inclination on specific contextual factor as well. In this section, the user-oriented Latent Dirichlet Allocation (LDA) and context-aware LDA models are first introduced. The corresponding Gibbs Sampling approach for inference of the model parameters and the recommendation query strategy definition are listed subsequently. Finally, the experiments conducted on the data provided by IP-based Television (IPTV) system “Vision” platform from Lancaster University confirmed that the extended context-aware model performs better on both accuracy and diversity evaluation metrics.

Symbols and their explanations used in this chapter are listed in Table 3.4. The user-oriented and the context-aware LDA share most symbols and definitions. Owing to the analogous data structure, we follow the classical symbols' definition in [131], and tune them slightly in accordance with our specific scenario.

Table 3.4: Symbols and notations for user-oriented LDA and context-aware LDA.

Symbols	**Notations**
$\alpha, \alpha^{c1}, \alpha^{c2}$	Symmetric hyper parameters, Dirichlet prior of $\vec{\vartheta}_u$, $\vec{\vartheta}_t^{tod}$, $\vec{\vartheta}_e^{type}$ respectively.
β	Symmetric hyper parameter, Dirichlet prior of $\vec{\varphi}_k$.
γ	Symmetric hyper parameter, Dirichlet prior of $\vec{\lambda}_u$.
φ_k	Programs' distribution w.r.t. topic k.
ϑ_u	Topics' distribution w.r.t. user u.
$\vartheta_t^{tod}, \vartheta_e^{type}$	Topics' distribution w.r.t. timing t and "live/VoD" condition e respectively.
λ_u	Propensity on three influential factors of user u.
$\underline{\phi}$, $\underline{\theta}$, $\underline{\lambda}$	Estimation of φ, ϑ, λ, respectively.
u, k, p, t, e	Index of users, topics, programs, time of day and environment(live or VoD) respectively.
U, K, V	Number of all users, topics and programs.
N_u	Number of transactions w.r.t. user u.
$v_{u,i}$	The i_{th} viewed item of user u.
$z_{u,i}$	Topic z, because of which user u chose the i_{th} item.
$s_{u,i}$	Path s, along which the topic $z_{u,i}$ is generated.
$t_{u,i}$	Time of day when recorded $v_{u,i}$.
$e_{u,i}$	Live/VoD environment of $v_{u,i}$.

3.3.1 User Oriented LDA Model

With the power of dimension reduction and hidden topic generation, topic models has been broadly attempted in both information retrieval and recommender fields. LDA is a well-defined generative topic model, which is originally proposed in the information retrieval area. As a Bayesian description framework, LDA models the probability distribution clustered on the hidden topics for texts. It can solve the problems as over fitting resulting from estimated parameters' linear growth and the unpredictability of unseen documents [70]. There have been multiple transformations of LDA applied in recommender systems [132–134], yet the basic idea behind these applications still heavily relies on the term-document matrix, i.e. the text description of items. Whereas in this section, we make use of user-program matrix to describe the generative procedure. Through the three-level hierarchical modeling procedure, topics' preference distribution of a specific user and the preference distribution of TV programs on specific topics are well represented in the graphical model and gains their specific probabilistic meanings.

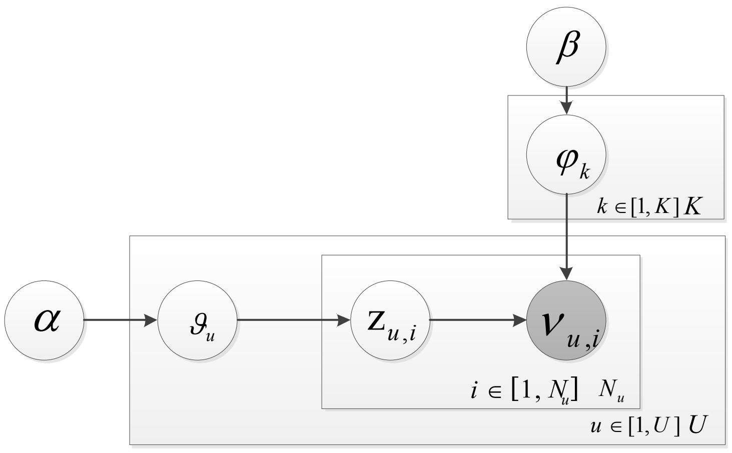


Figure 3.5: User-oriented LDA graphical model.

Given the similar structure of user-program matrix and document-term matrix, we apply LDA to infer users' preferences distribution on hidden topics. As depicted in Fig. 3.5, each user is treated as an outer plate and each transaction is treated as an inner plate. $v_{u,i}$, the only observed variable in the graphical model, represents the i_{th} item viewed by user u. $z_{u,i}$ indicates the topic assigned to $v_{u,i}$, which is also the sampling target in the sampling period. The frequency a user consumes a program serves as the sampling accordance in the LDA.

3.3.2 Context-aware LDA Model

On top of this user-based LDA, two contextual factors "live/VoD environment" and "time of day" are embedded in the model thus formulate the context-aware LDA. The influential roles of contextual factors are represented in the graphical model. From user-oriented LDA to extended context-aware LDA, we show the variation of probabilistic graphical models, and list the Gibbs Sampling formulas for both of them. In addition, corresponding query procedure is also presented. The experiments illustrate that context-aware LDA provides apparent performance gains on both ranking and diversity metrics compared to the user-based LDA.

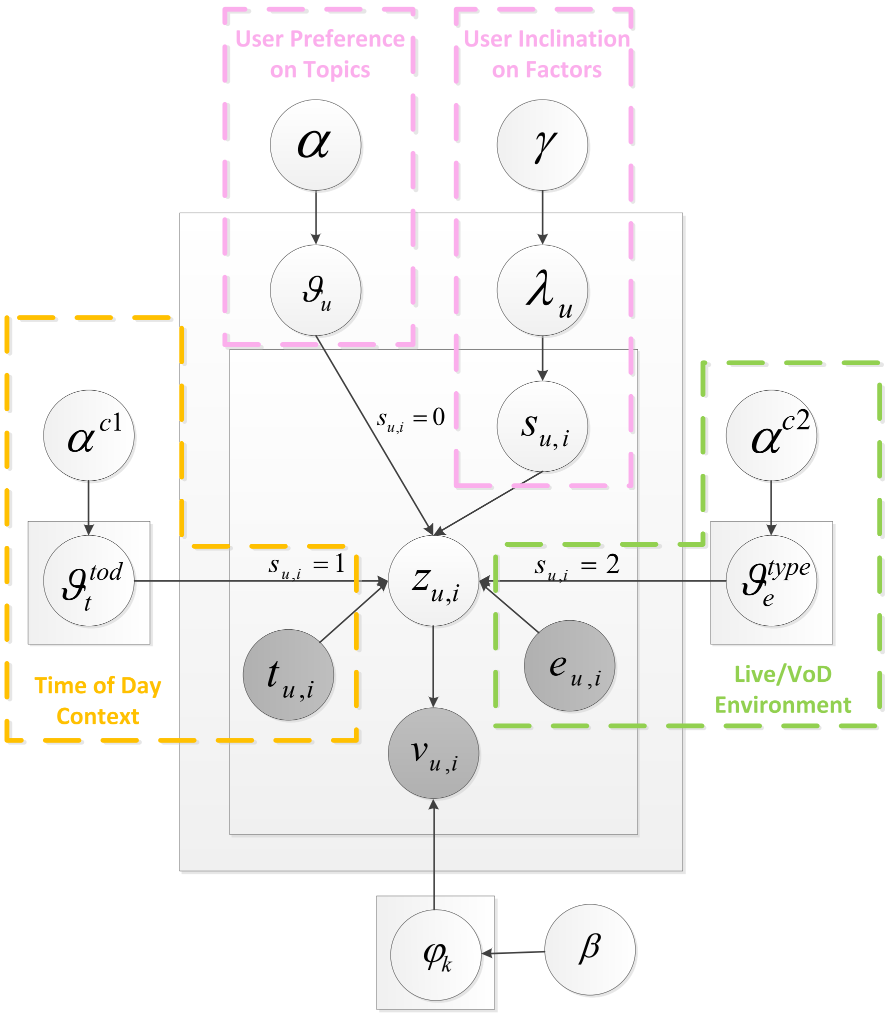


Figure 3.6: Context-aware LDA graphical model.

In the user-oriented LDA model, we only obtain users' individual preference, yet contextual variation cannot be reflected. Considering the extensibility of probabilistic graphical model [133], we extend the original LDA mentioned in last subsection to a context-aware LDA. Fig. 3.6 shows us the structure of this generating procedure. The probabilistic paths $\alpha \rightarrow \vartheta_u \rightarrow z_{u,i} \rightarrow v_{u,i}$ and $\beta \rightarrow \varphi_k \rightarrow v_{u,i}$ are the same as in user-based LDA, while the difference is reflected as the appearance of parts “User Inclination on Factors", “Time of Day Context" and “Live/VoD Environment". Among them, “user inclination" is the indicator that distinguishes users’ propensity on three factors: “his/her own preference", “time of day context" and “live/VoD environment". Since the dependency on personal preference and contextual factors can be individually different [13, 135], besides $z_{u,i}$, another hidden variable $s_{u,i}$ needs to be sampled in this model either. With the Dirichlet prior γ, $s_{u,i}$ can be sampled from multinomial distribution λ_u. It is clear from the graph that $s_{u,i} = 0$ determines $z_{u,i}$ generated from “User Preference" path, $s_{u,i} = 1$ points to the path of “Time of Day Context", while $s_{u,i} = 2$ matches the path of “Live/VoD Environment".

Due to the involvement of contextual factors “time of day" and “live/VoD environment", $v_{u,i}$ is no longer the only observed variable in the model. Timing indicator $t_{u,i}$ and environment indicator $e_{u,i}$ are also visible while sampling. When $s_{u,i} = 1$, the selection path goes to the options of “time of day” contextual factors, where $t_{u,i}$ will decide which ϑ_t^{tod} is to be used to generate topic $z_{u,i}$. On the other side, $s_{u,i} = 2$ means the path “live/VoD environment" context type is chosen and $e_{u,i}$ selects ϑ_e^{type} to sample $z_{u,i}$. In general, there are five hyper parameters: α, β, γ, α^{c1}(hyper parameter for context1) and α^{c2}(hyper parameter for context2) in this model. Mapping each of them, ϑ_u, φ_k, λ_u, ϑ_t^{tod} and ϑ_e^{type} are the distributions to be estimated. Considering the two invisible elements: $s_{u,i}$ and $z_{u,i}$, we need a two-step sampling procedure, which is introduced in next subsection [133].

3.3.3 Gibbs Sampling Formula

Gibbs Sampling is a high dimensional version of Markov Chain Monte Carlo Sampling. Its advantage is that given the full conditional probability, the acceptance rate can be 1 when sampling state on a single dimension, such that sampling effectiveness is maximized [136]. If we define K topics in the model, it means that K candidate states can be sampled for $z_{u,n}$ at each sampling, and there would be $\sum_{u \in [1,U]} N_u$ dimensions in total to switch at each iteration. The task of Gibbs Sampling is exactly iterative sampling for each $z_{u,n}$ until $\underline{\phi}$ and $\underline{\theta}$ reach a relative stable distribution.

In user-oriented LDA, four most important variables n_u (total number of transactions for user u), n_u^k (number of topics assigned with k w.r.t. user u), n_k (total number of transactions sampled as topic k), and n_k^v (total number of occurrences of program v assigned to topic k) contribute together with prior distribution α and β to the final full conditional probability [131]. Inferred by Bayes’ rule, this sampling probability is defined in Eq. 3.1.

$$
\begin{aligned}
& p(z_{u,i} = k | \vec{z}_{\neg(u,i)}, \vec{v}, \alpha, \beta) \\
& \propto p(z_{u,i} = k, v_{u,i} = p | \vec{z}_{\neg(u,i)}, \vec{v}, \alpha, \beta) \\
& = \hat{\vartheta}_{u,k} \cdot \hat{\varphi}_{k,p} \\
& = \frac{n^{k}_{u,\neg(u,i)} + \alpha_k}{\sum_{k=1}^{K}(n^{k}_{u,\neg(u,i)} + \alpha_k)} \cdot \frac{n^{p}_{k,\neg(u,i)} + \beta_p}{\sum_{v=1}^{V}(n^{v}_{k,\neg(u,i)} + \beta_v)}
\end{aligned}
\tag{3.1}
$$

However, in context-aware LDA, two-step Gibbs Sampling is needed to generate $s_{u,i}$ and $z_{u,i}$, respectively. Referring to the derivation in [137], the full conditional probability of path selection, i.e., the first step sampling, can be represented as in Eq. 3.2 - Eq. 3.4, where $n^{z_{u,i}}_{u,\neg(u,i)}$, $n^{z_{u,i}}_{t_{u,i},\neg(u,i)}$ and $n^{z_{u,i}}_{e_{u,i},\neg(u,i)}$ denote the number of topic $z_{u,i}$ regardless of element (u, i) w.r.t. user u, timing $t_{u,i}$ and live/VoD environment $e_{u,i}$ separately. By the same token, $n^{s=c}_{u,\neg(u,i)}$ indicates the number of path c excluding element (u, i) chosen by user u. Thus, together with Dirichlet prior α, α^{c_1}, α^{c_2}, β and γ, these counting numbers comprise the sampling formula for $s_{u,i}$.

$$
\begin{aligned}
& p(s_{u,i} = 0 | \vec{s}_{\neg(u,i)}, \vec{z}, \vec{v}, \alpha, \alpha^{c_1}, \alpha^{c_2}, \beta, \gamma) \\
& \propto \frac{n^{z_{u,i}}_{u,\neg(u,i)} + \alpha_{z_{u,i}}}{\sum_{k=1}^{K}(n^{k}_{u,\neg(u,i)} + \alpha_k)} \cdot \frac{n^{s=0}_{u,\neg(u,i)} + \gamma_0}{\sum_{c=1}^{3}(n^{s=c}_{u,\neg(u,i)} + \gamma_c)}
\end{aligned}
\tag{3.2}
$$

$$
\begin{aligned}
& p(s_{u,i} = 1 | \vec{s}_{\neg(u,i)}, \vec{z}, \vec{v}, \alpha, \alpha^{c_1}, \alpha^{c_2}, \beta, \gamma) \\
& \propto \frac{n^{z_{u,i}}_{t_{u,i},\neg(u,i)} + \alpha^{c1}_{z_{u,i}}}{\sum_{k=1}^{K}(n^{k}_{t_{u,i},\neg(u,i)} + \alpha^{c2}_{k})} \cdot \frac{n^{s=1}_{u,\neg(u,i)} + \gamma_1}{\sum_{c=1}^{3}(n^{s=c}_{u,\neg(u,i)} + \gamma_c)}
\end{aligned}
\tag{3.3}
$$

$$
\begin{aligned}
& p(s_{u,i} = 2 | \vec{s}_{\neg(u,i)}, \vec{z}, \vec{v}, \alpha, \alpha^{c_1}, \alpha^{c_2}, \beta, \gamma) \\
& \propto \frac{n^{z_{u,i}}_{e_{u,i},\neg(u,i)} + \alpha^{c2}_{z_{u,i}}}{\sum_{k=1}^{K}(n^{k}_{e_{u,i},\neg(u,i)} + \alpha^{c2}_{k})} \cdot \frac{n^{s=2}_{u,\neg(u,i)} + \gamma_2}{\sum_{c=1}^{3}(n^{s=c}_{u,\neg(u,i)} + \gamma_c)}
\end{aligned}
\tag{3.4}
$$

When the first step sampling is finished, the value of $s_{u,i}$, i.e., the path of generating procedure is determined. Given this $s_{u,i}$, the second step of topic sampling can be realized by the following posterior probabilities Eq. 3.5 - Eq. 3.7. Mapping with the contextual graphic model in Fig. 3.6, we can see how to discriminate these sampling formulas. Estimated users' preferences on topics $\hat{\vartheta}_u$ (calculated by $n^{k}_{u,\neg(u,i)}$) is multiplied when $s_{u,i} = 0$. Topics' trend estimation w.r.t. "time of day" $\hat{\vartheta}^{tod}_{t}$ (statistics on $n^{k}_{t_{u,i},\neg(u,i)}$) replaces this multiplier if $s_{u,i} = 1$. Under the condition $s_{u,i} = 2$, only $\hat{\vartheta}^{type}_{e}$ (estimation on topics' popularities regarding “live/VoD environment" using $n^k_{e_{u,i},\neg(u,i)}$) can be involved in the formula. In addition, $\hat{\varphi}_k$, i.e., the estimation of programs’ distribution on topic k, which is calculated by $n^v_{k,\neg(u,i)}$, is the common part for these formulas.

$$p(z_{u,i} = k | s_{u,i} = 0, \vec{z}_{\neg(u,i)}, \vec{v}, \alpha, \alpha^{c_1}, \alpha^{c_2}, \beta, \gamma) \propto \frac{n^k_{u,\neg(u,i)} + \alpha_k}{\sum_{k=1}^K (n^k_{u,\neg(u,i)} + \alpha_k)} \cdot \frac{n^{v_{u,i}}_{k,\neg(u,i)} + \beta_{v_{u,i}}}{\sum_{v=1}^V (n^v_{k,\neg(u,i)} + \beta_v)} \tag{3.5}$$

$$p(z_{u,i} = k | s_{u,i} = 1, \vec{z}_{\neg(u,i)}, \vec{v}, \alpha, \alpha^{c_1}, \alpha^{c_2}, \beta, \gamma) \propto \frac{n^k_{t_{u,i},\neg(u,i)} + \alpha^{c1}_{z_{u,i}}}{\sum_{k=1}^K (n^k_{t_{u,i},\neg(u,i)} + \alpha^{c2}_k)} \cdot \frac{n^{v_{u,i}}_{k,\neg(u,i)} + \beta_{v_{u,i}}}{\sum_{v=1}^V (n^v_{k,\neg(u,i)} + \beta_v)} \tag{3.6}$$

$$p(z_{u,i} = k | s_{u,i} = 2, \vec{z}_{\neg(u,i)}, \vec{v}, \alpha, \beta, \gamma) \propto \frac{n^k_{e_{u,i},\neg(u,i)} + \alpha^{c2}_{z_{u,i}}}{\sum_{k=1}^K (n^k_{e_{u,i},\neg(u,i)} + \alpha^{c2}_k)} \cdot \frac{n^{v_{u,i}}_{k,\neg(u,i)} + \beta_{v_{u,i}}}{\sum_{v=1}^V (n^v_{k,\neg(u,i)} + \beta_v)} \tag{3.7}$$

Having these sampling formulas, we can implement the Gibbs Sampling algorithms for both user-oriented LDA and context-aware LDA. Since sampling procedure of user-oriented LDA can be found as the same in [131, 138], we only focus on the algorithm for context-aware LDA (Alg. 1).

3.3.4 Evaluation Measurements

In this subsection, we list the convergence condition, and the metrics used in the experiments to evaluate the recommendation effect.

3.3.4.1 Perplexity and Convergence

We observe the convergence condition on the corpus to define whether the model training period is finished. *Perplexity* is a typical measurement in probabilistic graphical models to judge the convergence of the sampling process. The definition of perplexity is as Eq 3.8 shows. *M* always represents the model states (including estimated parameters) in the current iteration. In traditional term-documents LDA model, vector $\vec{v}$ denotes all the appearance of terms in the corpus. While in this thesis, program-user matrix is our main concern, thus $p(\vec{v_u}|M)$ tell the interactions of user u with all the programs in the dataset. The *perplexity* can be understood as the uncertainty of the model regarding the whole corpus. The lower the perplexity, the better model parameters fit the data set.

Algorithm 1: Gibbs Sampling procedure for the context-aware LDA.

```
Input: programs vector $\vec{v}$, hyper parameters $\alpha$, $\alpha^{c_1}$, $\alpha^{c_2}$, $\beta$, $\gamma$, topic number $K$,
       paths number 3
Output: paths assignments $\vec{s}$, topics assignments $\vec{z}$, estimations: $\underline{\phi}$, $\underline{\lambda}$, $\underline{\theta}$, $\underline{\theta}^{tod}$ and
        $\underline{\theta}^{type}$
//initialization;
for all users $u \in [1, U]$ do
    for all program-based transactions $i \in [1, N_u]$ do
        //sample path $s_{u,i}$ for $(u, i)$ $s_{u,i} = c \sim Mult(1/3)$;
        //sample topic $z_{u,i}$ for $(u, i)$ $z_{u,i} = k \sim Mult(1/K)$;
    end
end
//Gibbs sampling, burn-in period;
while not converged do
    for all users $u \in [1, U]$ do
        for all program-based transactions $i \in [1, N_u]$ do
            //sample path $s_{u,i}$
            $s_{u,i} = c \sim p(s_{u,i}|\vec{s}_{\neg(u,i)}, \vec{z}, \vec{v}, \alpha, \alpha^{c_1}, \alpha^{c_2}, \beta, \gamma)$;
            //sample topic $z_{u,i}$
            $z_{u,i} = k \sim$
             $p(z_{u,i}|s_{u,i} = c, \vec{z}_{\neg(u,i)}, \vec{v}, \alpha, \alpha^{c_1}, \alpha^{c_2}, \beta, \gamma)$;
        end
    end
    if converged then
        read out estimated distribution: $\underline{\phi}$, $\underline{\lambda}$, $\underline{\theta}$, $\underline{\theta}^{tod}$ and $\underline{\theta}^{type}$ ;
    end
end
```

$$perplexity(\vec{v}|\boldsymbol{M}) = exp - \frac{\sum_{u=1}^{U} \log p(\vec{v_u}|\boldsymbol{M})}{\sum_{u=1}^{U} N_u} \tag{3.8}$$

For user-oriented LDA and context-aware LDA, the calculation of $p(\vec{v_u}|\boldsymbol{M})$ may be slightly different. For user-oriented LDA, like Eq. 3.9 shows, values in probabilistic paths from user u to program v through K topics can directly sum up to $p(v|u)$ as $\sum_{k=1}^{K} \hat{\phi}_{k,v} \cdot \hat{\theta}_{u,k}$. Whereas for context-aware LDA, as depicted in the Fig. 3.6, aside from probabilistic paths passing through topics, selection paths s which decided by λ_u also need to be taken into account during topic selection. Thus in the Eq. 3.10, the probability that v appears in $\hat{v}_u$ is formed as the aggregation like $\sum_{s=1}^{3} \sum_{k=1}^{K} \hat{\phi}_{k,v} \cdot \hat{\theta}_{u,k}^{s} \cdot \hat{\lambda}_{u,s}$.

$$p(\vec{v_u}|\boldsymbol{M}) = \prod_{i=1}^{N_u} \sum_{k=1}^{K} p(v_{u,i}|z_i = k) \cdot p(z_i = k|u) = \prod_{v=1}^{V} (\sum_{k=1}^{K} \hat{\phi}_{k,v} \cdot \hat{\theta}_{u,k})^{n_u^v} \tag{3.9}$$

$$\begin{aligned} p(\vec{v_u}|\boldsymbol{M}) &= \prod_{i=1}^{N_u} \sum_{s=1}^{3} \sum_{k=1}^{K} p(v_{u,i}|z_i = k) \cdot p(z_i = k|u, s_i = c) \cdot p(s_i = c|u) \\ &= \prod_{v=1}^{V} (\sum_{s=1}^{3} \sum_{k=1}^{K} \hat{\phi}_{k,v} \cdot \hat{\theta}_{u,k}^{s} \cdot \hat{\lambda}_{u,s})^{n_u^v} \end{aligned} \tag{3.10}$$

3.3.4.2 Accuracy Oriented Metrics

Though accuracy measures make the recommender bias prone, they are still the important satisfaction measurement for judging the users' preferences.

nDCG@K has been introduced in the sub section 2.3.2. It is the normalized value of DCG_K (Discounted Cumulative Gain), which is used to measure the quality of the ranking. DCG_K is defined in Eq. 3.11, where for a ranked list, relevance of i_{th} element is symbolized as rel_i, while the penalizing factor $\log_2{(i+1)}$ weakens the relevance according to the ranked position. $IDCG_K$ presumes the most ideal condition that items in the top K list are ranked exactly as the same order as items ranked by their relevance descending. The DCG_K is calculated of this ideal condition. In this chapter, we utilize users' implicit feedback: playback percentage, as the relevance of item rel_i in a recommendation list.

$$\mathrm{DCG}_K = \sum_{i=1}^{K} \frac{2^{rel_i} - 1}{\log_2{(i+1)}} \tag{3.11}$$

$$\mathrm{nDCG}_K = \frac{\mathrm{DCG}_k}{\mathrm{IDCG}_k} \tag{3.12}$$

MRR (Mean Reciprocal Rank) is another measurement to judge the ranking lists, which focuses more on the first hit item in the recommendation list. As shown in Eq. 3.13, $rank_i$ represents the first hit item's position in the ranked list; the smaller the $rank_i$, the better the recommendation quality is. $|Q|$ denotes the number of queries in the test set, while on our test set it represents the total number of segments.

$$\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i} \tag{3.13}$$

Recall records the proportion of recalled programs among the user's viewed programs in the recommendation list. It reflects the coverage of the recommender. The definition can be traced back to sub section 2.3.2. Eq. 3.14 shows the definition regarding the corresponding formula.

$$\text{Recall@}K = \frac{\#\text{hit@}K}{\#\text{viewedprograms@}K} \tag{3.14}$$

3.3.4.3 Diversity and Novelty

The main purpose of introducing contextual factors in LDA is to provide more diversity in the recommendation list thus balance the bias caused due to the overemphasized accuracy. Therefore, evaluation measurements like diversity and novelty have been also included as the evaluation targets. We make use of the concept of diversity and novelty defined in [34] as the evaluation metrics.

Eq. 3.15 expresses the definition of **diversity**. More specifically, w.r.t. a specific user, L_{now} and L_{last} are recommended lists of arbitrary two neighbor segments. The difference set $L_{now} \backslash L_{last}$ holds the elements included in L_{now} yet not in L_{last}. With $Diversity(L_{now}, L_{last})@K$, only the top K elements will be remained in L_{now}, L_{last} separately. The size of the difference set of these two top-K lists, i.e. $|L_{now}@K \backslash L_{last}@K|$, divided by K is the diversity degree.

$$L_{now} \backslash L_{last} = \{x \in L_{now} | x \notin L_{last}\} \tag{3.15}$$

$$\text{Diversity}(L_{now}, L_{last})@K = \frac{|L_{now}@K \backslash L_{last}@K|}{K} \tag{3.16}$$

The **novelty** definition is similar to diversity to some degree. As shown in Eq. 3.17, $Novelty(L)@K$ can be seen as the top-K diversity between list L and historical list H, which is the cumulative set of seen items before L appears. $Novelty(L)@K$ stands for the proportion of the unseen items in the current recommendation list.

$$\text{Novelty}(L)@K = \frac{|L@K \backslash H|}{K} \tag{3.17}$$

3.3.5 Experiments

The experiments introduced in this section sets the goal to test the recommender effect of the newly proposed user-oriented LDA and context-aware LDA in the IPTV recommender scenario. From both accuracy and diversity perspectives, we investigate the bias balancing role of the proposed approaches.

3.3.5.1 Scenario and Dataset

Our recommender scenario is a campus IPTV provider from Lancaster University, which mainly serves university students with standard TV content. The dataset used in this section is comprised of one year transactions (from October 1th, 2013 to October 1th, 2014) recorded by the provider. We apply two criteria on the raw dataset to filter out some meaningless transactions:

- Playback duration less than 15 seconds and playback percentage less than 15%. A short viewing duration typically denote users' short hesitation on a program rather than their true interests. Only relatively stable viewing behavior is maintained to represent users' preferences.

- Users with total transaction number less than 10. Too small transaction number means low interactivity of a user. With the sparse interactions, it's difficult to effectively separate the training set, validation set and test set for those specific users.

After filtering, there are a total of 587 users and 119,435 transactions remaining. 16 genres, 33 channels and 7,065 different program names had been recorded among these transactions. Chronologically, the data was segmented into proportions 70% as training set, 20% as the validation set, and 10% as test set.

3.3.5.2 Candidate Recommendation Approaches

We evaluate the following five recommendation approaches (4 baseline approaches and a context-aware LDA) on the above mentioned dataset.

Rand assumes no patterns in user's preferences so that only randomly ranked programs are shown to users.

Hot selects programs with highest occurrence frequency in the training set and also ranks them according to their statistical popularity.

UserHot chooses the most frequently viewed programs w.r.t a specific user and also sorts them for this user by their frequencies.

User-oriented LDA as described in Subsection 3.3.1, deduce users' interests on the latent topics and quantify it as the probabilities.

Context-aware LDA as proposed in Subsection 3.3.2, distinguishes the users' preference under the context "time of day" or "live/VoD environment" besides users' personal preference.

3.3.5.3 Query Generation and Execution

Though the proposed context-aware LDA models the relation between user, context and latent topics well, to generate the recommendation strategy, the query generation strategy is still needed as the ranking accordance. Here we introduce the query generation and execution process based on the current PGM framework.

Query Generating - We define a segment as a time period within the same "time of day" (granularity of three hours) on a specific day for a user to distinguish query and recommending strategy. The final experiments will be done based on this unit of "segment" as well. In the test set, if there is only one program in the segment, solely users' preferences or contextual factors are used to predict this program, while in segment with multiple programs, the first program is treated as a hint to guess the following programs. More concretely, regarding algorithms user-oriented LDA and context-aware LDA, the generation of query would be different. Given the four possible combinations of algorithms and session forms, we could summarize the queries generating in Table 3.5. Here each sort of query $\vec{q}$ is represented as a vector of distribution on latent topics.

Table 3.5: Query vector generating.

Algorithms	**Segment with Single Program**	**Segment with Given Program** p
user-oriented LDA	$\vec{q}_u = \hat{\vartheta}_u$	$\vec{q}_{u,p} = \hat{\vartheta}_u \cdot \hat{\varphi}_{:,p}$
context-aware LDA	$\vec{q}_{u,t,e} = \hat{\lambda}_u * \left[\hat{\vartheta}_u; \hat{\vartheta}_t; \hat{\vartheta}_e\right]$	$\vec{q}_{u,t,e,p} = \vec{q}_{u,t,e} \cdot \hat{\varphi}_{:,p}$

Query Executing - Targeting on each query vector, we give the solution of similarity calculation, as listed in Table 3.6, to illustrate concrete approaches to execute query. In the test set, for the segment with only one program, where queries are $\vec{q}_u$ or $\vec{q}_{u,t,e}$, we utilize the probability of program v's appearance under different conditions, $p(v|u)$ or $p(v|u,t,e)$, as the similarity indicator. While for the segment with multiple programs, in which first program p is given, facing queries as $\vec{q}_{u,p}$ and $\vec{q}_{u,t,e,p}$, we calculate the cosine value between the query vector and $\hat{\varphi}_{:,v}$ to represent the similarity. The final ranked programs list for each segment is turned out through sorting by corresponding similarity.

Table 3.6: Query executing.

Query	**Score Calculation for Program v**
$\vec{q}_u$	$p(v\|u) \propto p(u,v) = \hat{\vartheta}_u * \hat{\varphi}_{:,v}$
$\vec{q}_{u,p}$	$\text{cosine}(\vec{q}_{u,p}, \hat{\varphi}_{:,v})$
$\vec{q}_{u,t,e}$	$p(v\|u,t,e) \propto \sum_c \lambda_c \cdot p(u,t,e,v\|s=c) = \vec{q}_{u,t,e} * \hat{\varphi}_{:,v}$
$\vec{q}_{u,t,e,p}$	$\text{cosine}(\vec{q}_{u,t,e,p}, \hat{\varphi}_{:,v})$

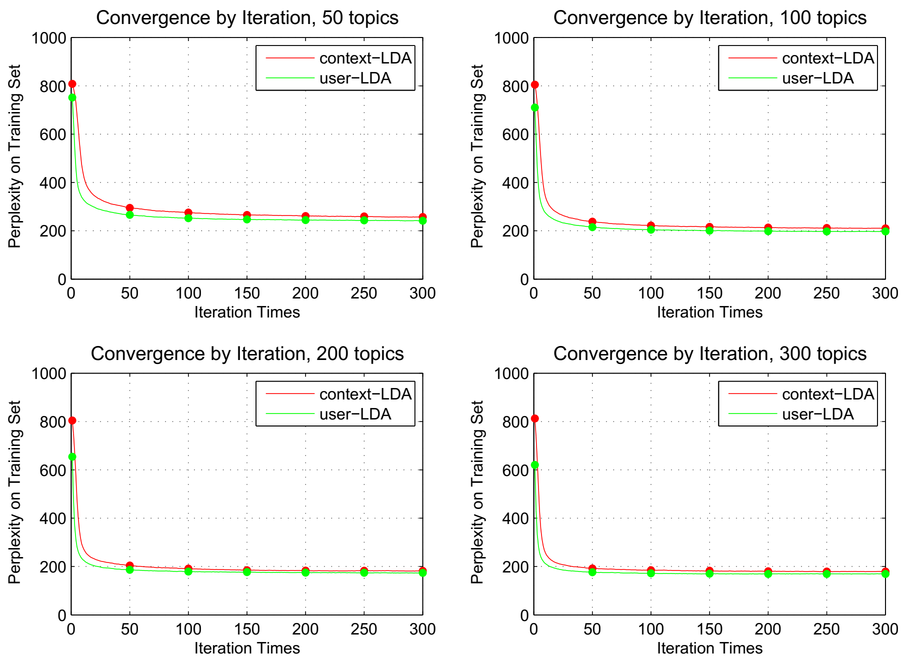


Figure 3.7: Convergence condition w.r.t. iterations. The number of topics has been chosen as 50, 100, 200, 300 respectively. The red curve represents the perplexity value for the context-aware LDA, the green curve denotes the user-oriented LDA.

3.3.5.4 Convergence Curve on Perplexity

As defined in the sub section 3.3.4, we use *perplexity* metric to judge the convergence condition of the model training for both user-oriented LDA and context-aware LDA. Fig 3.7 shows the convergence condition on perplexity regarding both user-oriented LDA and context-aware LDA with 50, 100, 150 and 200 topics respectively in training set. After 50 iterations, the model will tend to converge.

3.3.5.5 Evaluation Result

To evaluate the ranking quality of the candidate approaches, we compare them on “nDCG@K", “MRR@K" and “Recall@K". Generally speaking, when the topic number of context-aware LDA is over 35, it can almost outweigh all other approaches on these three metrics. On the other hand, when we take “Diversity" or “Novelty" also into account, apart from approach “rand", from which extremely disordered list would be turned out, context-aware LDA still shows good performance.

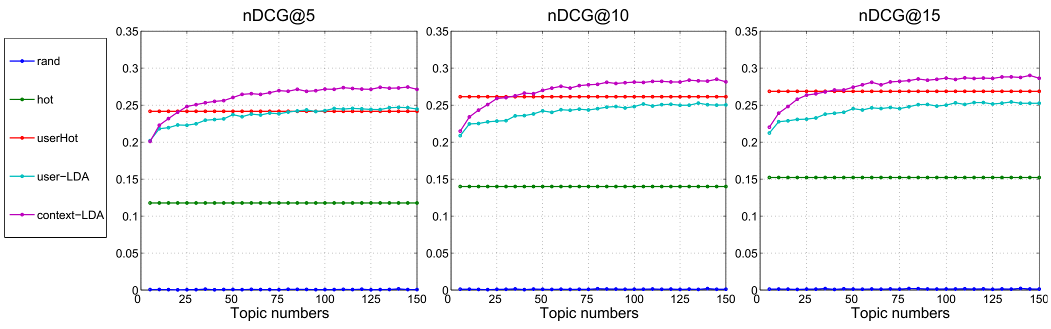


Figure 3.8: Performance comparison w.r.t. normalized Discounted Cumulative Gain. The “context-LDA" reaches the highest performance on nDCG when topic number $\geq$ 20, 35, and 40 for $K = 5$, 10, and 15 respectively.

Fig. 3.8 shows us the value of $nDCG@5$, $nDCG@10$ and $nDCG@15$ of all approaches w.r.t. different latent topic numbers, and the values shown on the figure are the mean $nDCG$ of all segments in the test set. Since the results of “hot" and “userHot" have nothing to do with topic numbers, they are just horizontal straight lines compared with results turned from LDA related approaches. Though “userHot" can outperform “user-oriented LDA" for certain settings, “context-aware LDA" is consistently the best of all the cases when the number of latent topics is set over 35.

As to the evaluation metric **reciprocal rank**, the performances of different approaches are depicted in the Fig. 3.9. MRR of “user-oriented LDA" is slightly higher than “userHot" after 40 topics, while “context-aware LDA" is always placed at first after 20 topics. The advantage of “context-aware LDA” is quite obvious in this case.

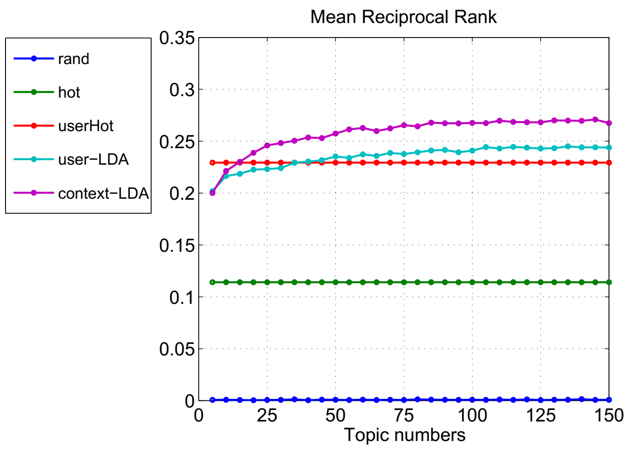


Figure 3.9: Performance comparison w.r.t. Mean Reciprocal Rank. The “context-LDA" outperforms the other approaches on MRR if the number of topics is greater than 15.

The comparison between the approaches on the **recall rate** is provided in Fig. 3.10. Due to the great performance of “userHot" on recall, “user-oriented LDA" behaves relatively weak and seems no longer competitive, whereas “context-aware LDA" can still maintain a satisfying result when the number of latent topics reaches 15 or 20.

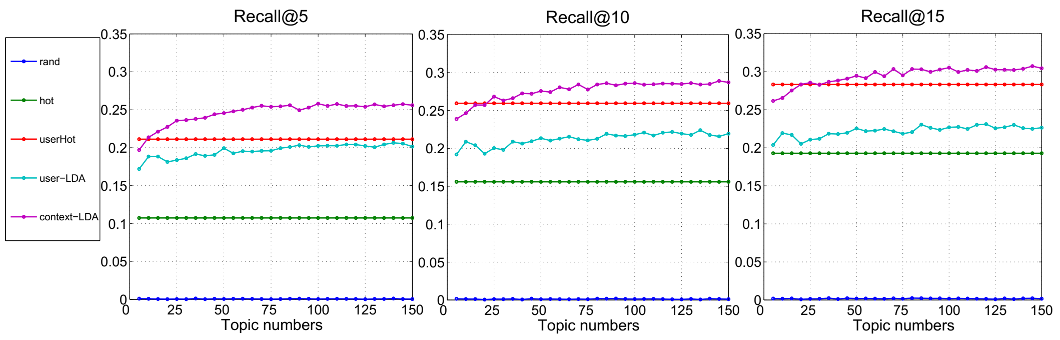


Figure 3.10: Performance comparison w.r.t. Recall Rate. The “context-aware LDA" outperforms other approaches on RR when topic numbers $\geq$ 10, 20, and 30 for $K = 5$, 10, and 15 respectively.

The purpose of involving contextual factors in the recommendation model is to increase the **diversity** and **novelty** in the recommendation lists and balance the biased condition of recommendations. The performances of the approaches concerning these two metrics are compared as well.

Since the performance curves are similar with different K concerning **diversity**, we only list $Diversity@10$ in Fig. 3.11. Though “rand" unsurprisingly gains the highest diversity owing to its extreme disorder, the absolute second place of “context-aware LDA" also illustrates the functionality of increasing diversity in recommended list of contextual factors involvement. Since “hot" and “userHot" always recommend same content, the diversities of them are all zero.

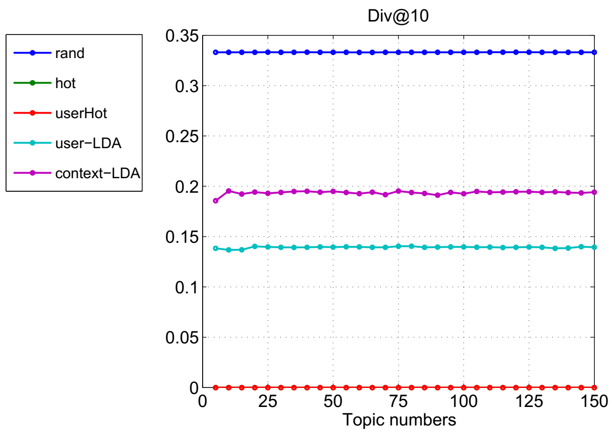


Figure 3.11: Performance comparison w.r.t. Diversity. As for Diversity, aside from the baseline “rand", the “context-LDA" approach shows dominant role, while the number of topics doesn’t make a big difference.

In addition, the **novelty** degree is evaluated in Fig. 3.12. However, regarding **novelty**, “context-aware LDA" does not exhibit so much dominant role as on other metrics; only narrow margin can be figured out over “user-oriented LDA".

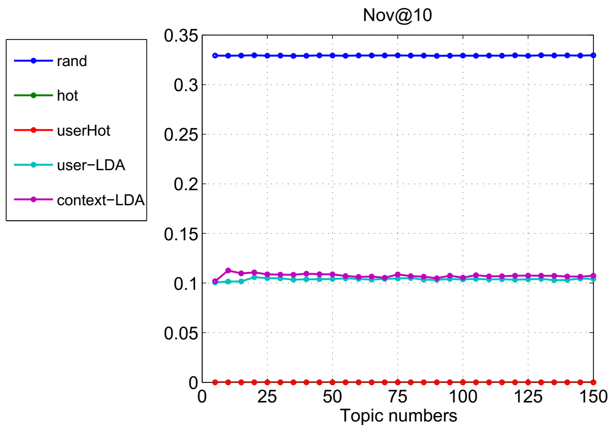


Figure 3.12: Performance comparison w.r.t. Novelty. Concerning Novelty, the “context-LDA" approach shows only a small advantage over “user-LDA".

3.3.6 Discussion

In this section, we solve the research problem **RP3.2: How do we model different contextual factors in one model and catch the corresponding users' inclination on the contextual factors?** Through applying LDA from IR domain to the RS domain, we make use of program-user matrix to build up the topic model. Based on user-oriented LDA, we add the *user's inclination path* between their personal preference and the contextual difference. With the contextual factors embedded in the PGM, we realized the goal of distinguishing users' individual inclination on the specific contextual factors. Theoretically, aside from "time of day", "Live/VOD" environment, any other contextual factors can be categorized and involved in this context-aware PGM. The evaluation result proves that context-aware LDA outperforms other benchmark approaches in both precision oriented and diversity oriented metrics. It balances the bias in the recommendation, while maintaining the high accuracy.

3.4 Countering Contextual Bias by Introducing Heterogeneous Resources

Though the trained recommendation model can predict user's preference well, it's inevitable that *filtering bubble* lead to biased recommendation result. In this section, we propose *contextual grid* to represent the *contextual bias* in IPTV system. To counter such bias in the IPTV system, we make use of external TV programs relevant social trend to give other reasonable yet diverse recommendations. To this end, we crawl the Twitter streams in the parallel time period as the IPTV user behavior data we got from the Lancaster University. We apply two trend indicators - TM and SigniScore in the Twitter stream for the attempt on capturing users' peak choices in the IPTV system. The experiment shows that both trend indicators captured in the external social sources won't contradict the internal user behavior, while provide more diverse options.

3.4.1 Trends and Peaks Definition for Breaking Events

Having the temporal altering frequency distribution of user request data and tweets data at hand, quantifying the definition of *trend* and *peak* is critical for comparing the information of internal and external data sources. *Peaks* in IPTV system for a specific program are collected based on the users' choices on that program. The trending hotness of topics for Tweet streams are represented as the appearance frequency of specific keywords combination from the programs' titles [139, 140].

For the Tweets data, a rising phase of a *peak*, say *trend*, foretells the popularity of a program in the near future. *Trend* indicators in program relevant Tweets possess the capability

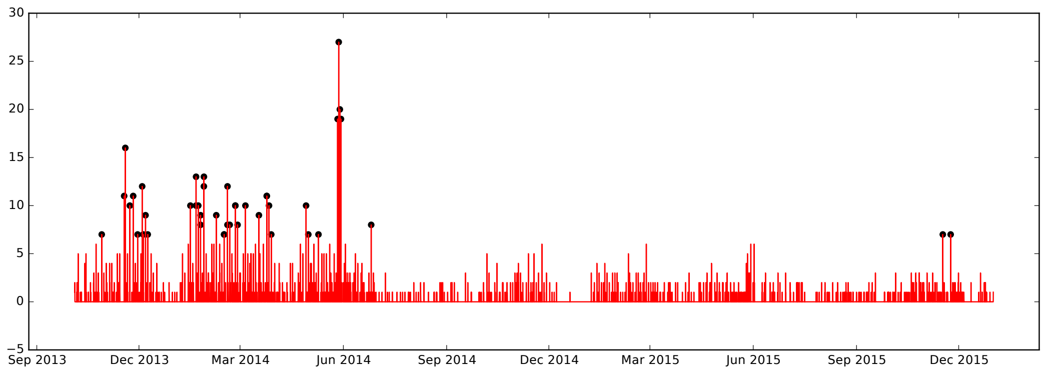


Figure 3.13: Peaks detected in the corresponding user request.

of early detection on popularity *peaks* in both Twitter text stream and heterogeneous IPTV user requests. To verify this assumption, we clarify the definitions of *peak* and *trend* in the text stream, which is a topic of active research. We use two indicators known from stock market analysis – Trend Momentum [141] and SigniScore [142] to capture the hotness of specific TV programs in the crawled Tweets data.

In the user request data, we are interested in the *peak* moments which constitute the exact moments of high interest for the chosen program. We define the *peak* duration of specific programs as the time bins containing significant higher number of user requests compared to the average number over the full period of discrete time-bins. In accordance with standard notions of significance in the literature, the threshold is set at two standard deviations, i.e., time bins with more than $\mu + 2\sigma$ user requests are defined as *peaks*. We present a complete timeline for a program, which is with 1-hour time bins (as Figure 3.13). Its corresponding *peak* moments are marked as solid circles at the frequency tops of Figure 3.13.

$$MA(n,k) = \frac{\sum_{i=n-k+1}^{n} x(i)}{k} \tag{3.18}$$

$$MACD(n) = MA(n, k_{\text{fast}}) - MA(n, k_{\text{slow}}) \tag{3.19}$$

$$TM(n) = MA(n, k_{\text{fast}}) - MA(n, k_{\text{slow}})^{\alpha} \tag{3.20}$$

$$Momentum(n) = MA(TM(n), k_{\text{smooth}}) \tag{3.21}$$

The first *trend* indicator used in this thesis is the Trend Momentum (TM) score listed in Eq. 3.20, which is proposed in [141]. It is a smoothed version of the Moving Average Convergence-Divergence (MACD) stock trend indicator and has been deployed to detect trending news in the Twitter stream. The definition stems from the concept of Moving Average (MA) (Eq. 3.18), which captures the average frequency of k previous time bins at the n_{th} time bin. Considering that this average is not enough to represent a rising or decreasing trend, MACD as shown in Eq. 3.19 utilizes the difference between the MA over k_{fast} (fewer) time bins and the MA over k_{slow} (more) bins to determine whether there is a trend. Additionally, a discount parameter α is assigned as an exponential decay term to the longer period in MACD, which defines TM in Eq. 3.20. The sign change of its value from negative to positive or vice versa indicates the appearance of a rising or declining trend. Furthermore, to avoid a volatile condition, MA is applied again with a third, even shorter time window k_{smooth} to smooth the trend indicator, it is presented in Eq. 3.21. Throughout the rest of this thesis, we simply use the name TM to refer to the final *Momentum* value (Eq. 3.21). With the concept of momentum, a trend is said to be emerging when there is a turning point from negative to positive. There are four hyper-parameters (k_{fast}, k_{slow}, k_{smooth} and α) can be tuned to improve the accuracy of predicting *peaks*.

$$\Delta \leftarrow x - EWMA \tag{3.22}$$

$$EWMA \leftarrow EWMA + \alpha \cdot \Delta \tag{3.23}$$

$$EWMVar \leftarrow (1-\alpha) \cdot (EWMVar + \alpha \cdot \Delta^2) \tag{3.24}$$

$$\alpha = 1 - \exp\left(\frac{\log(0.5)}{t_{half}}\right) \tag{3.25}$$

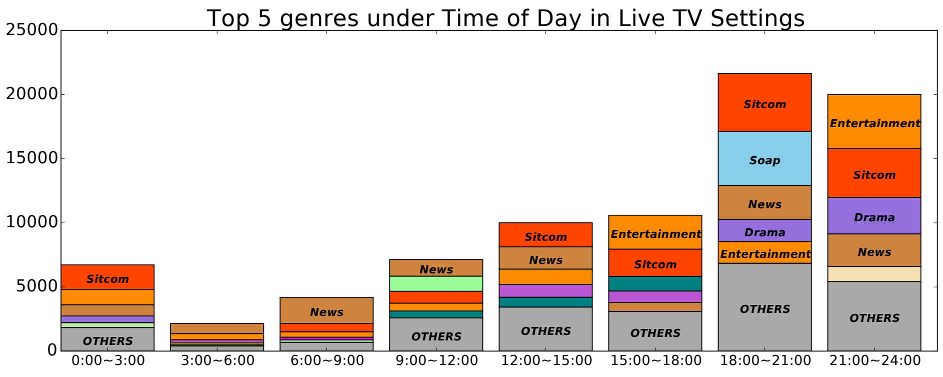
$$\text{SigniScore}(x, \beta) = \frac{x - \max(EWMA, \beta)}{\sqrt{EWMVar} + \beta} \tag{3.26}$$

Another trend indicator called *SigniScore* is defined in Eq. 3.26 [142]. It is also a member of the MA family. With x being the frequency of occurrence inside a time bin, the definition of Δ in Eq. 3.22 represents the deviation of this time bin from Exponentially Weighted Moving Average (EWMA) calculated over preceding bins. With the current time observed time bin, Δ is added to the EWMA in Eq. 3.23. Here α is used as a weighting factor, similar to a learning rate. Corresponding to the accumulated mean EWMA over the frequency stream, the formula for the moving variance is given in Eq. 3.24. As shown in Eq. 3.25, α is computed from the half-life time t_{half}, which depends on the application domain and must be adjusted according to expert knowledge. In our case, the critical parameter t_{half} is used as one of the hyper-parameters to be optimized. On top of EWMA and Exponentially Weighted Moving Average Variance (EWMAVar), *SigniScore* is defined in Eq. 3.26 in the form of *z-score*. β is the bias term that avoids division by 0 and at the same time filters noise. It constitutes the second hyper-parameter which will be searched for regarding *SigniScore*. *SigniScore* gives the significance degree of the trend measurement, thus makes different trending moments comparable. On the other hand, the threshold crossings (around 0) is set as the trend indication signal.

3.4.2 Contextual Bias Representation

In the “Vision” system , TV programs broadcast as live TV, it is also recorded simultaneously by a cloud-based service for VOD retrieval. Users decide by themselves how and when to retrieve arbitrary content. We show users’ biased choices on both live TV and VOD system. To reveal the typical “time of day” *contextual bias* in the user behavior, we split a day by grouping it into three hours long segments, such that eight time segments span a whole day. We observe and analyze the biased condition targeting *genres* and *programs* separately. Both *statistical representation* and *contextual grid representation* reflect the *contextual bias* intuitively.

3.4.2.1 Statistical Contextual Bias Representation



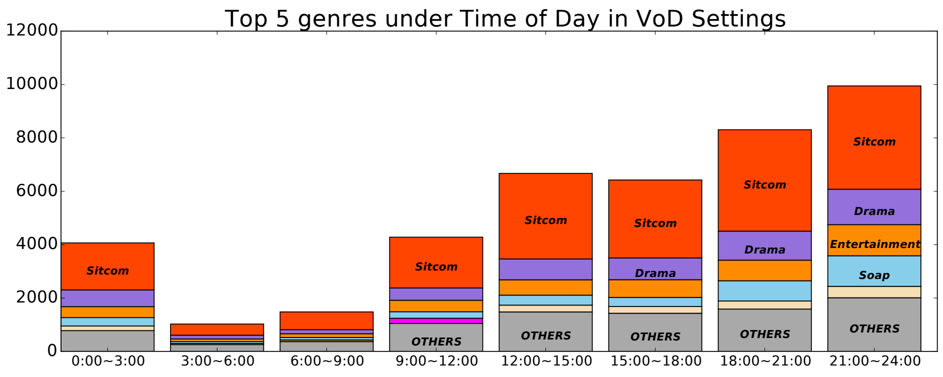


Figure 3.14: Frequency distribution on **genres** per “time of the day” in Live and VOD modes. In the Live mode, *News* receive more attention in the morning time, while *Entertainment*, *Sitcom* and *Drama* gain the most views in the afternoon and late night. While in the VOD mode, Top3 genres rank consistenty between the time segments, the bias shows stronger pattern in the VOD scenario.

By observing the frequency distribution of “genres” and “programs” over the eight time groups in Figure 3.14 and Figure 3.15, we see that the biased preference for certain types or program titles is much stronger in the VOD scenario compared to that in linear TV. In Figure 3.14, we rank the viewing frequency regarding genres as a stacked bar chart (genres below the fifth rank are colored gray) for different time segments. Two user consumption modes – Live and VOD – are shown in the upper and lower sub-figure separately. Under live mode, in the morning (03:00 to 09:00), *News* broadcasts receive more attention, while *Entertainment*, *Sitcom* and *Drama shows* constitute the most dominant content types consumed in the afternoon (15:00 to 18:00) and late night (21:00 to 24:00). *Soap* contributes a lot to the evening time period (18:00 to 21:00). In the VOD mode, the Top 5 genres (*Sitcom*, *Drama*, *Entertainment*, *Soap*, *Comedy*) rank consistently between time segments. The selection bias within user choices is stronger in the VOD scenario.

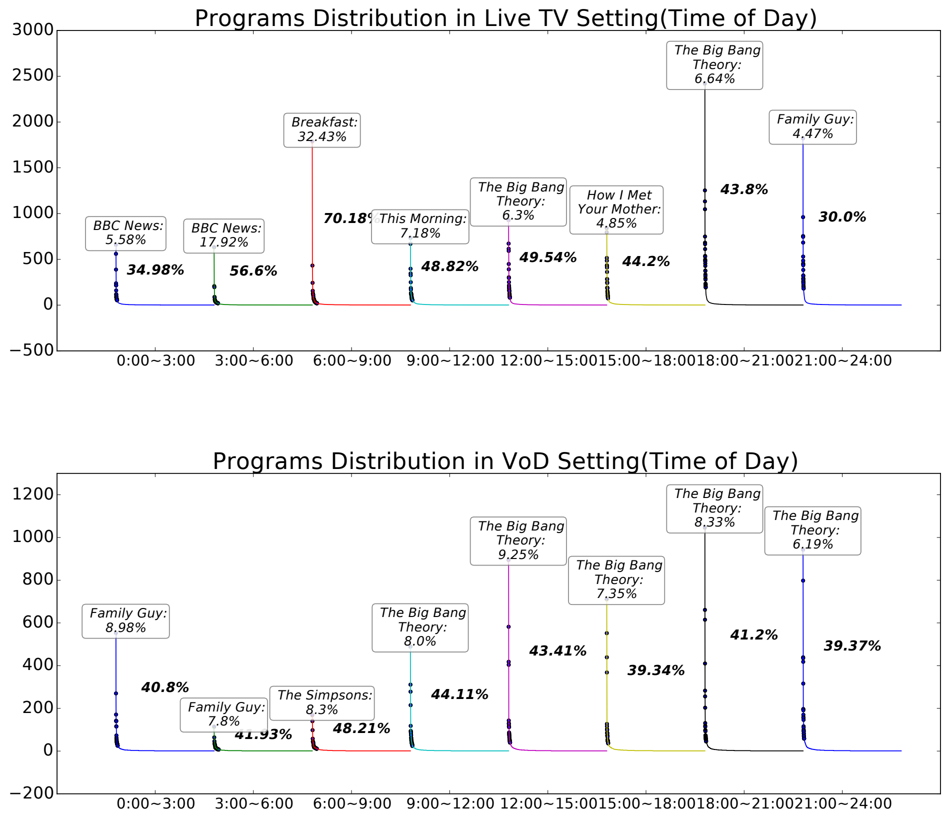


Figure 3.15: Frequency distribution on **programs** per "time of the day" for Live and VOD modes. The frequency distribution of programs shows a power law distribution in each time segment. The most viewed program title and its proportion in a time segment is specified in the corresponding circle. Top 30 out of 12,809 unique programs occupy the dominant 30% to 70.18% views across the time segments.

On a smaller granularity, the frequencies of the Top 30 most viewed programs are displayed as black dots for each time segment in both live and VOD settings in Figure 3.15. The most viewed program title and its proportion in this time category is specified next to its corresponding circle. In addition, the percentage of the top 30 programs in each time segment is also written as text in the middle of every sub-figure. The frequency distribution of programs shows a power law distribution in each time segment. Out of 12,809 unique program titles, the proportion occupied by the Top 30 programs in each time segment ranges from 30% (21:00 - 24:00 in live settings) to 56.6% (3:00 - 6:00 in live settings), with an outlier of 70.18% (6:00 - 9:00 in live settings). It appears that a short head (30/12,809 0.23%) of the programs attracts around 30% to 50% of user playback requests within every "time of day" category. It suggests that *contextual bias* exists not only in high-level user selections (such as genres) but also at the program-level.

3.4.2.2 Contextual Grid Representation

Having computed the contextual bias from the statistical perspective, we consider the bias problem in the view of a single TV program. We propose to use *contextual grid* to represent the contextual distribution of peaks/trend frequency for a specific program. A *contextual grid* is a 2-dimensional grid recording the entries mapping "time of day" and "day of week" segments combinations. The grouping frequencies of *peaks* and *trends* are collected within the corresponding context cells. Theoretically the grid can be extended to any dimensions with contextual factors.

Entropy value H of the head distribution inside each *contextual grid* is a good measurement as contextual bias (the lower the entropy, the more skewed conditions in the grid distribution, and the higher the bias). For instance, Eq. 3.27 describes the entropy calculation for user request *peaks* concerning a specific program. Since we draw the grid from dimensions "time of day" and "day of week", the subscript (i, j) indicates an entry index (i_{th} interval on the "time of day" axis and j_{th} day on the "day of week" axis) in the grid $\mathbf{G}$. Superscript on $\mathbf{G}$, like $peaks$ for $\mathbf{G}^{peaks}$ and TM for $\mathbf{G}^{TM}$, is used to distinguish the different indicators under which the grid values are generated. The frequency distribution $p(i, j)$ is then calculated for contextual entry empirically as $\frac{\#\{\text{indicators} \in \mathbf{G}_{(i,j)}\}}{\#\{\text{indicators} \in \mathbf{G}\}}$, and the superscript on p also tells the indicator used to generate the grid. A lower entropy value means more certainty about the user behavior and thus the more bias. Therefore, directly comparing entropy values is one method to measure the bias degree.

$$H(\mathbf{G}^{\text{peaks}}) = - \sum_{(i,j) \in \mathbf{G}^{\text{peaks}}} (p^{\text{peaks}}_{(i,j)} \log{(p^{peaks}_{(i,j)})}) \tag{3.27}$$

Take an example for the program *Coronation Street*, a British Soap Opera is visualized in Figure 3.16. The heatmap on the left hand side shows a contextual grid $\mathbf{G}^{peak}$ on user request *peaks*. The heatmap on the right hand side shows the contextual grid $\mathbf{G}^{signi}$ on SigniScore crossing points using the Twitter source. The grid entries with higher count number are encoded in red. Within the IPTV system, user choices for the program *Coronation Street* are only distributed between 18:00 - 21:00 on Mondays, Wednesdays and Fridays, which coincide with the airing schedule. The corresponding entropy value H over the grid $\mathbf{G}^{peak}$ is accordingly lower. However, for the trend signals of this program detected in the parallel Twitter stream, the heated grids distribute broader, thus leads to a higher H value for the grid $\mathbf{G}^{signi}$and lower bias degree.

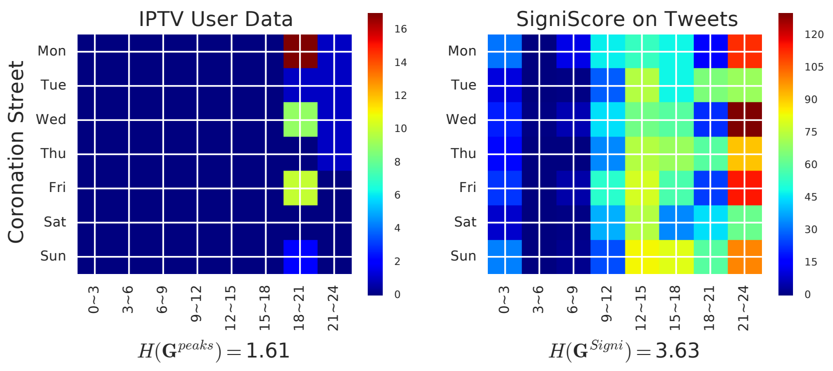


Figure 3.16: Contextual grid example for TV program *Coronation Street*.

The entropy value comparison directly reflects whether the new indicator has the debiasing effect over user request *peaks*. In addition to the direct comparison on H value between two grids, dis-similarity between the distributions can also present the difference even the bias degrees are similar. We choose Jesen-Shannon Divergence (JSD), a symmetric divergence based on Kullback-Leibler Divergence (KLD) to illustrate such difference between the two frequency distributions on two grids. The KLD defined by $D_{KL}(P \parallel Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$ (as in Eq. 3.28) is a measure of how one probability distribution P diverges from another probability distribution Q, or the information gain achieved if P is used instead of Q. Since the value of KLD is not symmetric for two given distributions P and Q, the intermediate distribution M, mean of of P and Q ($M = \frac{1}{2}(P + Q)$ in Eq. 3.29) turns out a symmetric measure. Thus JSD [143], which is the mean of $D_{KL}(P \parallel M)$ and $D_{KL}(Q \parallel M)$ as shown in Eq. 3.30 provides a symmetric measure of the divergence between the distributions over different *contextual grids*.

$$D_{KL}(P \parallel Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)} \tag{3.28}$$

$$M = \frac{1}{2}(P + Q) \tag{3.29}$$

$$JSD(P||Q) = \frac{1}{2}\left(D_{KL}(P \parallel M) + D_{KL}(Q \parallel M)\right) \tag{3.30}$$

3.4.3 Social Trend as Heterogeneous Resource

In order to counter the *contextual bias* exists in the user behavior inside the internal IPTV system, we propose introduce the external social resources to serve as an additional recommendation option. Fig. 3.17 describes the basic idea of introducing social trend as external contextual factor to IPTV systems. For multiple program titles, we track the users' viewing frequency and crawl the corresponding Tweets through Twitter "keyword" search. After grouping the frequency data into 1-hour time bins, trend indicators (TM [141] and *SigniScore* [142]) calculated on Twitter data can signal the proper recommending moment for a specific program. Through the timing offset of trending moments comparison in Tweets and moments of peaking interest in the user behavior data tracking, we show that a heterogeneous social resource doesn't necessarily cause deviation from the users' own preferences and can be considered as a valid external contextual factor for a recommender.

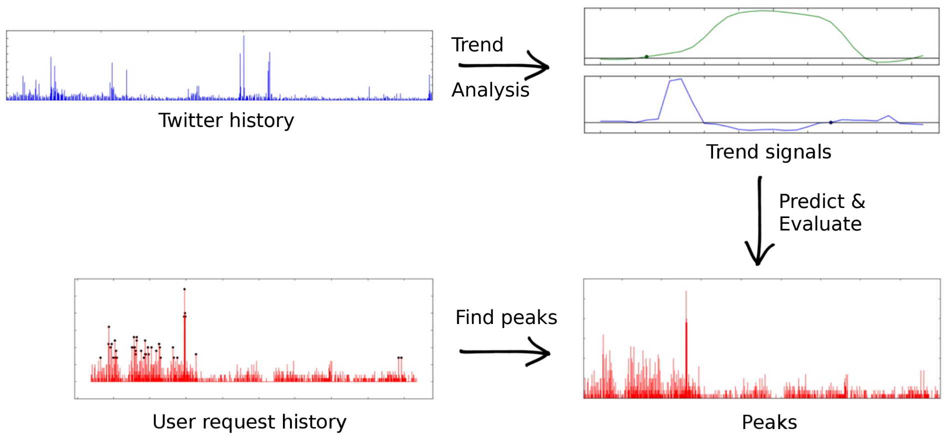


Figure 3.17: Trending events in Twitter as external contextual factor to provide recommendation moment.

Twitter is one of the most popular social media platforms for second screen usage in the UK. Therefore, we opted the Tweet Stream as the external contextual resource to address the issue of users' selection bias in the IPTV system. Since a program's title is the finest granularity item attribute in the TV platform, we obtain our data by crawling the tweets only referring to relevant program titles. To narrow the investigation range, we select twelve titles from the 30 most requested TV programs on the "Vision" platform and performed a Twitter keyword search [1] to crawl related Twitter histories. In total, the crawler gathered 4.7 million tweets for the twelve TV programs across a 26 month period parallel to the user request dataset. We group tweets corresponding to the program titles into fixed interval (i.e., one hour) and use the frequency distribution along the time axis to represent the program-relevant on Twitter.

3.4.4 Experiment and Evaluation

To verify that the proposed schema (introducing trend information in Twitter stream to serve as recommendation indicator) is able to counter the *contextual bias* without violating the users' preference much, we set up the experiments and evaluation in this subsection.

3.4.4.1 Experimental Setup

We obtain TV program related Tweets posted within a time span parallel to the user request data in the IPTV system "Vision" (Oct. 2013 to Dec. 2015, in total 26 months). Given the fact that "Vision" is a UK-based TV platform, for the 12 targeted TV titles, we choose ten UK productions and two US productions as candidate programs. In Table 3.7, for both user request data and the crawled Twitter data we provide the number of data points per program as N and group these data points into one hour time bins. Among these time bins, *peaks* are the ones where the number of tweets/requests exceeds two standard deviations over the mean per (non-empty) time bin in the respective data stream. Following this definition, the user dataset contains a total of 464 *peaks* for the targeted twelve TV programs, while the comparably bigger Twitter dataset contains 3,455 such *peaks*. Though there are relatively few peaks in the user request dataset, the generally low values of δ_{12} (the number of times where two consecutive peaks appear within a range of twelve hours) exclude the risk that they all belong to a few major *peak windows*.

3.4.4.2 Consistency between External Trends and Internal Peaks

In accordance with definitions in the literature [141, 142], we declare the crossing point when a trend measure crosses a threshold to be the signal for a rising trend in the data stream. We evaluate the precision of using *trends* on Twitter to predict *peaks* in user request data by two scores: OR and earliness of signal Δt. If the closest threshold crossing point occurs after the nearest peak, we count the trend prediction as *missed*, contributing to a higher OR. Otherwise, the crossing point can be seen as a successful indication of the incoming *peak*. An example can be found in Fig. 3.18 where TM successfully predicts

[1] https://twitter.com/i/search/

	User request data			Twitter data	
	N	#peaks	δ_{12}	N	#peaks
Made in Chelsea	1235	20	0	649523	111
EastEnders	1195	38	1	1276494	310
Hollyoaks	2070	91	13	1010904	375
Gogglebox	911	28	0	502954	128
Match of the Day	1122	23	2	189331	275
Emmerdale	1601	67	1	457419	175
Coronation Street	2624	43	7	295738	221
Britain's Got Talent	683	13	3	126100	102
Frasier	1413	72	23	116387	531
North West Tonight	1338	20	0	76512	430
This Morning	1046	21	0	17415	223
My Wife and Kids	756	28	6	10115	574

Table 3.7: Comparative statistics per evaluated program for user requests and Twitter data. N is the number of data points. When grouped by 1 hour, Peaks are the bins with more than $\mu + 2\sigma$ data points. δ_{12} counts the number of times where two consecutive peaks appear within a range of 12 hours.

the *peak*, while SigniScore misses the chance because of the late arrival of its threshold crossing signal. For every successfully predicted trend indicator, we compute the time delay between the threshold crossing and the nearest *peak* point as Δt, which shows how early a Twitter stream *trend* can predict user choice *peaks* (just as Δt shown in Fig. 3.18). The threshold can later be used to optimize towards precision or recall. In our evaluation we use for simplicity a fixed threshold $\theta = 0$ for finding crossings. In addition to the two trend measurements, we include a baseline trend indication in the evaluation. The baseline indication directly uses the *peaks* in the Tweet stream as trend indication signal, where the threshold is again two standard deviations over the mean. The calculation of the baseline is straightforward and no parameters need to be tuned.

The performance of a trend indicator in a particular domain depends on the values assigned to its hyper-parameters. We deploy grid search to find the best hyper parameter settings for our target application. The performance criterion to be minimized is the average number of missed peaks (OR), over the eight target programs. Since all preliminary trials reported the best value of α to be 0.8 for TM, we keep it constant throughout optimization. Subsequently we conduct a 3-dimensional grid search with ranges $k_{slow} \in [2, \ldots, 48], k_{fast} \in [2, \ldots, 36], k_{smooth} \in [1, \ldots, 18]$, skipping trials where $k_{fast} > k_{slow}$. For the 2-dimensional heatmaps as displayed in Fig. 3.19 we keep k_{smooth} fixed at six and k_{fast} at 18 respectively as they achieved the lowest omission rate throughout the full

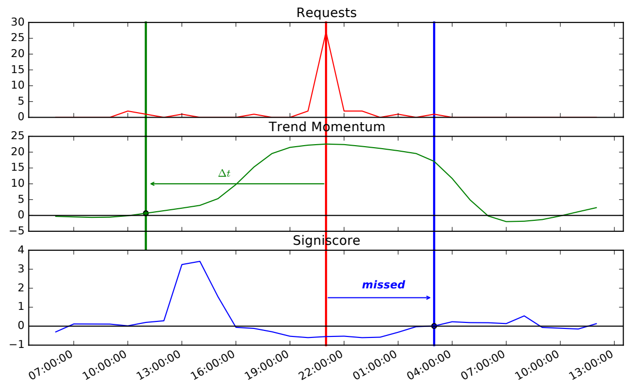


Figure 3.18: Visual comparison of a successful trend indication in TM versus a *miss* in *SigniScore*.

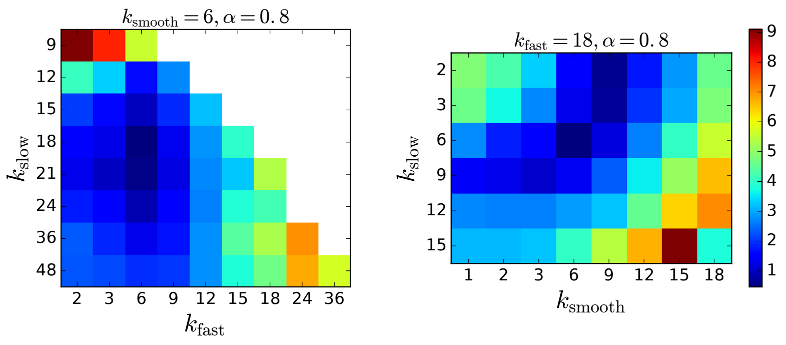


Figure 3.19: Omission rate heat map for TM. Under combinations of k_{fast}, k_{slow}, and k_{smooth}, the heat map shows the omission rate (in %) respectively.

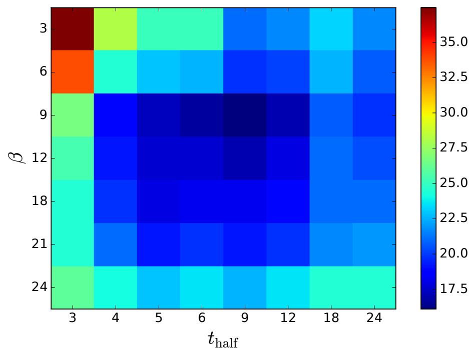


Figure 3.20: Omission rate head map for *SigniScore*. For various combinations of t_{half} and β, the heat map shows the omission rate (in %).

search. The parameters settings falling in the dark blue area are better for reducing the omission rate. The results indicate that the best parameter settings for TM are $[k_{slow} =$

$18, k_{fast} = 6, k_{smooth} = 6, \alpha = 0.8]$. As to *SigniScore*, the bias term β and half-life setting t_{half} are the parameters to be tuned. Since there are only two parameters to be searched, a single grid is used in the range as depicted in Figure 3.20. The results show that $[\beta = 9, t_{half} = 9]$ is the optimal parameter combination for the problem at hand.

The best hyper parameters determined by the grid search are used to examine the per-program performance. In addition to the OR, for each program, we compute the average Δt, i.e., the mean time difference between threshold crossing and closest *peak*, to denote how early the trend indicator signals the incoming burst in interest. Aside from that, the standard deviation σ is also provided for Δt. Contrary to the usual case where trend detection is favored to be as early as possible, in our case, lower delays represents higher correlation between trend signal in tweets and *peaks* in user data. Thus we consider a small Δt (and small $\sigma(\Delta t)$ accordingly) to be better.

	Trend Momentum			SigniScore			Baseline		
	OR	Δt	σ	OR	Δt	σ	OR	Δt	σ
Made in Chelsea (20)	**5.0%**	**11.0**	1.5	**5.0%**	14.3	4.8	15.0%	3.9	1.6
EastEnders (38)	**2.6%**	**8.7**	2.1	7.9%	10.7	9.5	26.3%	11.3	15.1
Hollyoaks (91)	**3.3%**	8.0	2.6	20.9%	**6.6**	3.2	31.9%	17.6	23.4
Gogglebox (28)	**3.6%**	7.6	1.8	7.1%	**7.0**	2.0	35.7%	69.1	15.9
Match of the Day (23)	**0.0%**	**4.4**	2.3	39.1%	20.0	14.6	39.13%	33.9	14.2
Emmerdale (67)	**1.5%**	8.2	2.2	9.0%	**6.6**	2.8	49.3%	112.5	122.5
Coronation Street (43)	**2.3%**	9.8	2.5	25.6 %	**6.2**	4.0	51.2%	31.5	25.9
Britain's Got Talent (13)	30.8%	**9.4**	2.0	**7.7%**	10.7	9.5	100.0%	-	-
Average	3.7%	8.3	-	16.1%	7.9	-	39.9%	39.8	-

Table 3.8: OR, delta time duration Δt and the standard deviation σ have been listed for each indicator. The number of peaks per program is displayed in braces after the TV program name. TM shows better performance on OR than Signiscore and Baseline.

Among the twelve programs for which we crawled Twitter data, we evaluate the eight target programs separately from the rest (Table 3.8 and Table 3.9 respectively). The results for the suitable candidates reveal that TM outperforms *SigniScore* and *baseline* in terms of OR, Δt (also considering $\sigma(\Delta t)$). The threshold crossing shows a relatively consistent value for Δt before *peaks* of about eight hours. This early increase in communication is observed for all soaps and reality shows (like *EastEnders* and *Gogglebox*) in the dataset. To a lesser extent, sports programs such as *Match of the Day* very well reflect a general characteristic of Twitter second screen usage. The reality TV show *Britain's Got Talent* constitutes the only case where *SigniScore* achieves lower OR, but its data is particularly sparse with only 13 peaks in total. Another special case is marked by *Made in Chelsea*, for which the *baseline* already shows reasonable performance ($\Delta t = 3.9$), whereas both trend measurements cross the threshold much earlier. Generally speaking, the low average OR of the two trend indicators (3.7% for TM and 16.1% for SigniScore) proves the high consistency between user request data and external social stream data for the target programs. Thus, in situations where user interest is peaking, a recommendation based on the external trend information will not deviate from the user preference in the internal IPTV system. Nevertheless, as to the question whether *trend* indicators have the alleviating effect on contextual bias in IPTV systems, it will be presented in the next subsection.

	Trend Momentum			SigniScore			Baseline		
	OR	Δt	σ	OR	Δt	σ	OR	Δt	σ
My Wife and Kids (28)	21.4%	7.8	6.9	25.0%	12.5	10.4	25.0%	106.8	220.6
Frasier (72)	47.2%	13.3	12.4	40.3%	1.6	1.8	45.8%	19.4	20.9
This Morning (21)	95.2%	17.0	0.0	61.9%	153.1	80.5	57.1%	11.6	8.5
North West Tonight (20)	70.0%	10.0	2.7	50.0%	476.5	335.8	60.0%	87.4	75.7
Average	52.48%	11.3	-	41.84%	77.1	-	45.39%	49.4	-

Table 3.9: Evaluation of trend indicators and baseline for the excluded programs. The number of peaks per program is displayed in braces. TM, SigniScore and Baseline all show high omission rate.

The varied evaluation results over different programs suggest that the applicability of the trend indicator as a forecast for bursts in interest depends on the specific program. For the other four programs out of twelve candidates, the performance as displayed in Table 3.9 sets a negative example of the trend indicators' applicability for forecasting. Considering programs like *This Morning* and *North West Tonight*, their term collocations are not unique, thus the crawler tends to retrieve a significant number of unrelated Tweets, which in turn adds noise to the trend signal, hampering its usefulness. Similarly expected is the failure in the case of US programs like *My Wife and Kids*, for which the matching between Twitter trends and program interest is skewed due to different time zones.

3.4.4.3 Bias Countering Effect on Contextual Grid

Having confirmed the consistency between external social source and internal user request data, we now address the question whether the external resource can counter contextual bias detected in the IPTV system. We analyze such effect on *contextual grids* proposed in subsection 3.4.2.2 w.r.t. two points: 1) pair-wise dissimilarity of frequency distributions on different *contextual grids*, which shows the deviation degree of specific *trend* indicators; 2) comparison of the entropy values inside each *contextual grid* heat map, which directly reflects the difference on bias degree.

First, the Jensen-Shannon Divergence JSD between two contextual grids is calculated

Program	Peaks/TMs	Peaks/SigniScores
Gogglebox	0.69	0.50
Coronation Street	0.69	0.51
Made in Chelsea	0.69	0.62
Match of the Day	0.69	0.62
Hollyoaks	0.69	0.52
Emmerdale	0.68	0.52
EastEnders	0.69	0.56
Britain's Got Talent	0.66	0.53

Table 3.10: Jensen-Shannon Divergence between the Grids of different indicators w.r.t. candidate TV Programs.

as a measure of difference between two indicator frequency distributions over grid of chosen contextual factors. JSD can be understood as the symmetric version of the KLD. The higher the JSD, the more different the two frequency distributions are. As listed in Table 3.10, targeting the eight candidate programs, we calculate such divergence correspondingly. Generally, the JSDs between *peaks* in user request data and TM over the domain relevant Tweet stream range from 0.66 to 0.69. They are slightly higher than the distribution difference measured between *peaks* and SigniScores (ranges from 0.50 to 0.62). That is to say, considering a specific TV program, if we only think about the dis-similarity of recommendation offers from external source, TM can better foretell users' general interests.

The degree of bias with respect to contextual factors can be expressed in the *contextual grid* as entropy value. As proposed in Section. 3.4.2.2, we take two contextual factors, "time of day" and "day of week", to analyze the bias condition. The result is a two dimensional grid with "time of day"-groups of three hours on the x-axis and "day of week" cells on the y-axis. To measure the contextual distribution of *peak* and *trend* indicators, we colorize each of the respective contextual groups according to the number of indicators, resulting in a heatmap. In Figure 3.21, we draw the *contextual grid* for three indicators. $\mathbf{G}^{\text{peak}}$ represents the *peaks* in user request, $\mathbf{G}^{\text{TM}}$ stands for threshold crossings of *Trend Momentum* caught on Tweets and $\mathbf{G}^{\text{Signi}}$ denotes the *SigniScore* in Tweets. 5 TV programs *Match of the Day*, *Hollyoaks*, *Emmerdale*, *EastEnders* and *Britain's Got Talent* have been chosen as the examples with the representation of the contextual grid, each of them lay at a specific row. The corresponding entropy value over each *contextual grid* is calculated and shown below each grid. Apparently, for the specific example programs, the entropy value grows from $H(\mathbf{G}^{\text{peaks}}) \in [1,2]$ to $H(\mathbf{G}^{\text{TM}}) \in [2,3]$, and again up to $H(\mathbf{G}^{\text{Signi}}) \in [3,4]$.

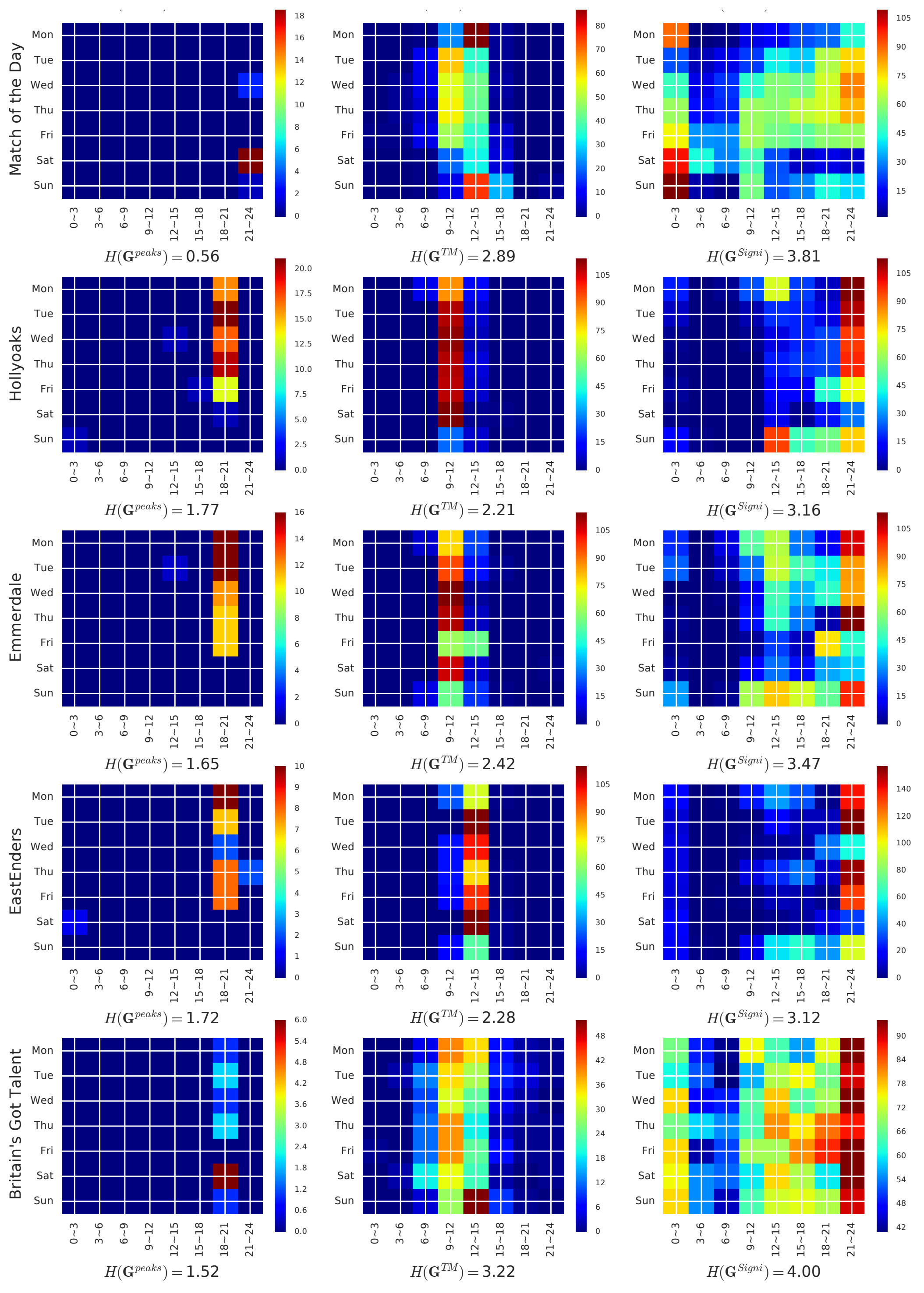


Figure 3.21: Heat map of the frequency distribution over *contextual grid*. 5 representative TV programs have been tracked on the three indicators *IPTV user*, *Trend Momentum on Tweets* and *Signiscore on Tweets*. $H(\mathbf{G}^{\text{Indicator}})$ denotes the entropy of the specific indicator on this grid. The lower the entropy value, the higher the biased condition.

The higher the entropy value, the more uniformly the frequency of the indicators is spread over the grid, and the stronger the contextual de-biasing effect. Based on the consistent high value of Signiscore behaves in the entropy, we conclude that SigniScore is a more suitable trend measure to be applied for contextual de-biasing in the TV domain.

3.4.5 Discussion

In this section, we focus on solving the research problem **RP3.3: How to counter contextual bias in TV recommendation scenario?** *Contextual grid* and external resources involvement are the main research contributions. We analyze user interactions in a hybrid VOD/linear TV platform and identify a prevalent *contextual bias*. Consumers' choices tend to fall in a strong temporal regularity, only a few dominant programs or channels account for the majority of consumption at certain times. Such contextual bias is represented through the *contextual grid* w.r.t. a specific TV program.

We harvest Twitter as a source of external social context to counter this contextual bias. The concepts of *peak* and *trend* are applied in both internal and external stream data. *Trend Momentum* and *SigniScore* as *trend* indicators are attempted in the experiments. Both of them show good performance as early signals for population-wide bursts of interests (i.e. *peaks*). Being compared on a *contextual grid* in terms of entropy, the higher entropy acquired by SigniScore, the more potential for de-biasing IPTV user behavior.

3.5 Conclusions

In this chapter, we thoroughly investigate the role of contextual factors in recommender system in the scenario of IPTV system. Questionnaire feedback and focus group discussions bring the insight for modeling the contextual factors and countering the contextual bias in the work later on. In general, timing, location, social influence and breaking news are all notable contextual factors. The necessity of discriminating individual difference on the inclination of different contextual factors and involving breaking domain relevant news/events is also put forward from the questionnaire result.

Though the better recommendation effect has confirmed the meaning of modeling contextual factors in the context-aware LDA, the representation of *contextual bias* both the statistic result and the *contextual grid* format still exhibit the inevitable *filtering bubble* in the user requests. To counter such kind of bias, we propose to incorporate external social stream on relevant TV programs. On the crawled Twitter stream and IPTV user data, trend indicators like TM and SigniScore are used to foretell the *peaks* in the user request data and serve as an alternative recommendation indicator. Experiment and evaluation illustrate that both trend indicators work well for the *peaks* moment forecasting in user request data, while SigniScore outweighs TM in the de-biasing effect.

CHAPTER 4

NEWS RECOMMENDER: TOPIC LEVEL UNDERSTANDING OF POPULARITY BIAS

Item-level granularity analysis is conventional in recommender systems, especially in entertainment-focused scenarios like IPTV systems and news recommenders. However, content-level level understanding plays an important role for providing interpretable recommendations. Especially in the digital news portals, users consume the events and topics discussed in the articles rather than solely individual items. Thus, the definition and the analysis of the bias in news recommender systems on the text or topic level attract a lot research attention these years. In this chapter, we analyze the news recommendation scenario using the data of the NewsREEL challenge (News REcommenataion Evaluation Lab). The clicks pattern analysis for the popular articles confirms its predicting power and benefit for winning the challenge. Based on the pattern, we propose use Topic Concern Degree and Topic Ranking Bias to track the content level ranking bias. The ranking bias from both temporal and behavioral views present the special characteristics for online publishers.

4.1 Introduction

News, as the most important media content, has the role of guiding social opinion. The perception of the world is highly influenced by virtual social networks and the items we read. Digital news publishers provide users with convenient access to articles online to alleviate them from searching for preferred articles. Many publishers deploy recommender systems or similar personalization techniques to assist users in finding relevant news quickly and conveniently. Different from general item-level recommendation scenarios (e-commerce and portfolio recommendation), what users consume in *digital news* *portals* are the topic or event of news articles. However, topics and events are not like concrete features or attributes (like color, size of items), they are comprised of terms and named entities, which needs to be extracted carefully. Extracting these topics and events are vital for tracking users' preference on the content-level. With the extracted topics or events, measuring the hotness and the bias degree also deserves research attention.

There are three ways to incorporate recommenders into digital news portals. First, as an *e-Magazine Provider*, *Flipboard*[1] aggregates news content from different third party providers and selects news which is relevant to a user's pre-defined topics to form their personalized news board. Second, several *Content Providers*, such as *ByteDance*[2], generate news contents themselves. They compute recommendations in a closed system based on internal users, news, and interaction in between both. Third, *Recommendation Providers*, e.g. *plista*[3] and *outbrain*[4], offer recommendation services for different kinds of news websites. Table 4.1 compares characteristics of the three main-stream news recommenders concerning aspects such as whether they generate content by themselves, the stability of users, and the fixed source of news items.

Table 4.1: Characteristics of main-stream news recommender.

News Recommender	Generating Content	Stability of Users	Fixed Source of News
e-Magazine Provider (e.g. Flipboard)	✗	✓	✗
News Content Provider (e.g. Bytedance)	✓	✓	✓
Recommendation Provider (e.g. PLISTA)	✗	✗	✗

As a representative of *Recommendation Providers*, PLISTA manifests its non-trivial recommendation scenario in terms of variety of news portals and differences in users' expectations. To continuously explore good recommendation strategies, PLISTA held the NEWSREEL challenge for 4 years starting from 2013 [108]. The participants' tasks are recommending news articles for different news publishers. The solutions developed by the challenge participants must fulfill the following requirements:

- recommendations must be provided within 100ms upon request;
- participants need to deal with news portals from different domains;
- the user groups within a specific portal change;

[1]https://flipboard.com/
[2]https://www.toutiao.com/
[3]https://www.plista.com/
[4]https://www.outbrain.com/

- number of messages varies largely among portals [144, 145]. In contrast to recommending movies or music, news items continuously emerge and become outdated constituting a dynamic environment.

The leader board of NEWSREEL online challenge in the first three years showed that popularity based algorithms always win. Winners of NEWSREEL2016 presented their ensemble recommendation strategy including only popularity-based approaches in [146,147]. Their approach occupied the dominant first place for the whole online evaluation period. In the same year, authors of [18] discovered that the power law distribution of frequency on impressions and clicks can be caught even hourly for some specific publishers.

In this chapter, we study how to explain the success of popularity-focused recommenders in the NewsREEL scenario. Following RPs are targeted within these studies.

RP 4.1: To what extent is the predicting power of the popularity bias for the CTR of the news recommender? CTR is the most frequently used commercial evaluation criteria for a news recommender. In order to increase the chance that the recommended items get clicked, the user behavior like *impression* and *clicks* are tracked in the news recommender. The popularity bias collected from users' skewed choices usually presents its own prediction power on CTR. In a competition, like the NEWSREEL challenge, the algorithms that make use of the popularity bias ranked higher. Therefore, the correlation between the popularity bias and the predicting power is proposed to be studied first.

RP 4.2: How to parse and analyze the topics with respect to the popularity bias in the news consumption stream ? We ascribe the reason of the dominant role of popularity based approaches in news recommenders to the continuity of topic concern bias. To analyze such topic ranking bias effect from different dimensions, the concrete parsing approach on the topics concern and ranking bias are interesting research problems to be solved.

RP 4.3: How could we interpret the topic ranking bias from temporal view and user behavioral view? In most research work, the understanding of bias is in the static mode through out the whole dataset. However, both topic concern bias and topical ranking bias are supposed to change dynamically. How to track the evolution of the bias to make better use of them is rarely researched. With the quantified topic concern and ranking bias, the cross prediction effect between different user behaviors also deserves attention. Therewith, we analyze the ranking bias from both temporal and user behavioral view.

In order to solve the above RPs, the research work have been conducted accordingly. The contributions of this chapter can be included as : 1) we summarize the successful popularity-based approaches in the NewsREEL challenge and observe the influence window of the bias. On the *article level*, the prediction power of such popularity bias is measured with Jaccard Similarity. 2) The definitions of Term Concern Degree (TCD) and Topic Ranking Bias (TRB) are proposed on the basis of term frequency and ranking similarity. TCD denotes the hotness or the degree of the popularity bias, while TRB tracks this hotness of multiple biased topics in the ranking mode. 3) The topic level bias is analyzed in both temporal and behavioral modes. The pattern differences captured in different news portals are also presented.

The sections in this chapter are organized as following: Section 4.2 describes our participation in NewsREEL2016 and records the relevant findings from the popularity bias and the its predicting power. Thereafter, the definitions of the topic concern degree and topic ranking bias are proposed in Section 4.3. Subsequently, Section 4.4 discovers such ranking bias difference in different news portals from both temporal view and user behavior view. Finally, Section 4.5 concludes the chapter and gives an outlook to additional research fields.

4.2 Popularity Bias and Predicting Power

The NewsREEL challenge provides researchers with the opportunity to evaluate their news recommendation algorithms live based on real users' feedback. Since 2014, participants evaluated a variety of approaches on the Open Recommendation Platform (ORP). The popularity-based algorithms constitute the most successful ones. In this section, we chronologically recorded our participation in NewsREEL online task in the year 2016. Through deploying approaches including "most impressed", "newest", "most impressed by category", "content similar" and "most clicked", it turns out that purely dealing with clicking information already possesses quite good CTR performance. More specifically, for the portal *Sport1*, the extrapolation of the time series of impressions and clicks enables us to predict the items most likely to be clicked in the next hours. A sample analysis on one week data shows us that the duration of an item being popular is much longer than we expected.

4.2.1 Approaches used in NewsREEL

In this section, we chronologically record our deployed approaches in NewsREEL2016 as following. As the comparison result, the popularity-based approaches perform well in the participation duration and raise our attention to the research on the popularity pattern in news recommendation in the following sections.

4.2.1.1 Most Viewed

Inspired by the good performance of "baseline" in the past years (see [148]), which directly uses the most recently viewed items as recommendation candidates, we implemented a similar approach which sorts the 2000 most recent impressions by their frequencies. Typically, this approach is called "most popular", but to distinguish it from "most clicked" which will be introduced later on, we refer to it as “most viewed” here. The approach ran on ORP for two weeks (January 31 to February 13, 2016), and got the CTR 1.21% (ranked 3rd, team “artificial intelligence” got the first place with CTR 1.48%) and 1.35% (ranked 2nd, team “abc” got the first place with CTR 1.4%) in these two weeks separately.

4.2.1.2 Newest

Considering that freshness represents a vital aspect of news, we also implemented an approach “newest” which provides the most recently created items from the same category as recommendation. Given the good performance of “most viewed” mentioned above, we use “newest” as an alternative solution when the request lacked an item_id, i.e. the category cannot be determined. In addition, for a recommendation request with 6 candidate slots, 3 positions are still filled by “most viewed”. Therefore, this approach can be seen as a simple ensemble of “most viewed” and “newest”. With this solution, from 21–27 February, our team “news_ctr” got CTR 1.19% (ranked 5th, team “artificial intelligence” got the first place with CTR 1.45%) in the contest leader board.

4.2.1.3 Most Viewed by Category

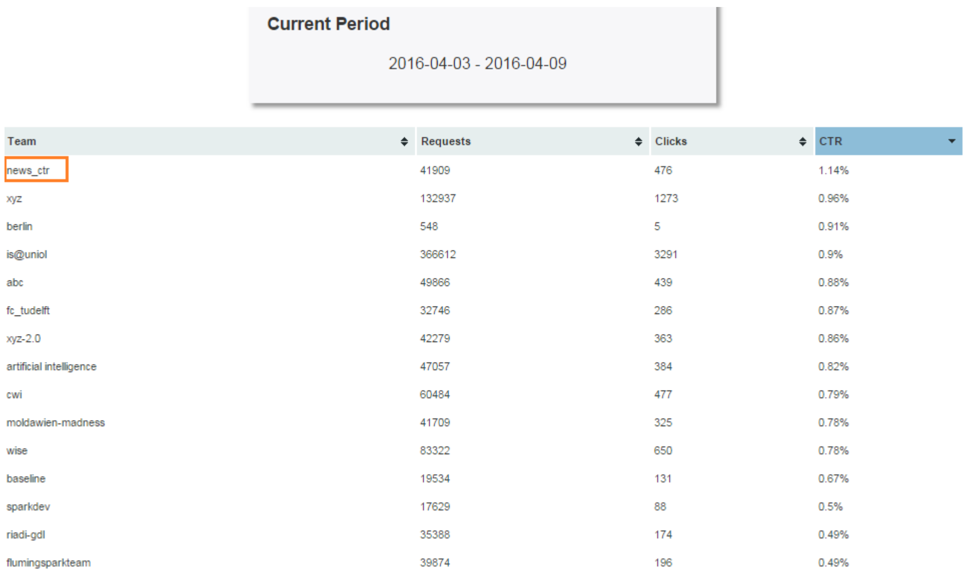
After witnessing how “newest” weakened the effect of “most viewed”, we conducted another experiment which only considered the number of impressions, while the impression counts are separated by categories. For the recommendation request with item_id, the recommending targets are the “most viewed” items in the same category as the item_id acquires. The approach ran on ORP for three weeks, from 6–12 March 2016 it got CTR of 0.82% (ranked 7th, team “is@uniol” got the first place with CTR 1.03%), from 13–19 March 2016 it got CTR of 0.97% (ranked 11th as the last one on the board, team “xyz” got the first place with CTR 1.85%) and from 20–26 March 2016 it got CTR 1.24%(ranked 6th, team “xyz” got first place with CTR 2.16%).

4.2.1.4 Content Similarity

Knowing the worse performance of category integration in the impression popularity approaches, we implemented a pure content-based recommender which suggested recommended items according to document similarity generated from Apache Lucene index. This content-based recommender was deployed on ORP from 27th March to 2nd April and got CTR of 0.77% (ranked 11th, team “xyz” got the first place with CTR 1.51%). The result shows that pure content similarity is not sufficient for a successful recommending strategy. The results consolidate the findings from Said et al. [149], that content similarity fails to pick up on new stories.

4.2.1.5 Most Clicked

Though contest teams used different algorithms, the clicked items for all of the recommendations tended to be similar. This consistent regularity reminded us to think whether characteristic patterns exist within clicks along the time axis. Hence we implemented the simplest approach "most clicked" which only serves the most frequently clicked items in the last hour to the recommendation requests. From 3–9 April 2016, this simple approach got a CTR of 1.14% winning the leader board ahead of "xyz" (0.96%). Figure 4.1 shows the results during this week.



Current Period

2016-04-03 - 2016-04-09

Team	Requests	Clicks	CTR
news_ctr	41909	476	1.14%
xyz	132937	1273	0.96%
berlin	548	5	0.91%
is@uniol	366612	3291	0.9%
abc	49866	439	0.88%
fc_tudelft	32746	286	0.87%
xyz-2.0	42279	363	0.86%
artificial intelligence	47057	384	0.82%
cwi	60484	477	0.79%
moldawien-madness	41709	325	0.78%
wise	83322	650	0.78%
baseline	19534	131	0.67%
sparkdev	17629	88	0.5%
riadi-gdl	35388	174	0.49%
flumingsparkteam	39874	196	0.49%

Figure 4.1: *news_ctr* got first place in the period 3–9 April, 2016.

4.2.1.6 Discussion

Having observed the outperformance of the popularity in "clicks", we looked into previous research works related to the contest. Kliegr and Kuchař [150, 151] implemented an approach based on association rules. They used contextual features (e.g. `ISP`, `OS`, `GEO`, `WEEKDAY`, `LANG`, `ZIP`, and `CLASS`) to train the rule engine. However, these association rules do not outperform other algorithms. Through the investigation in [152], Gebremeskel and de Vries found that there is no striking improvement including geographic information on news recommendation, yet more randomness of the system should be taken into account. Doychev et al. introduced their 6 popularity-based and 6 similarity-based approaches in [153], but their algorithms performed poorly compared with "baseline" due to being influenced by content aggregating. Said et al. concluded in [149] that news article readers might be reluctant to be confronted with similar topic all the way, but more pleased to be distracted by something breaking or interesting.

Both former research and the approaches tried in year 2016 indicate that pure document similarity or contextual information involvement introduce worse CTR. Pure clicks popularity based approaches predict better on the readers' near future clicks preferences.

4.2.2 Clicks Pattern Analysis

Being aware of the high influence of "clicks" in the recommendation effect, we analyze the clicks behavior in the two dominant portals - *Sport1* and *Tagesspiegel* on April 5, 2016 to explore the special patterns in the log data.

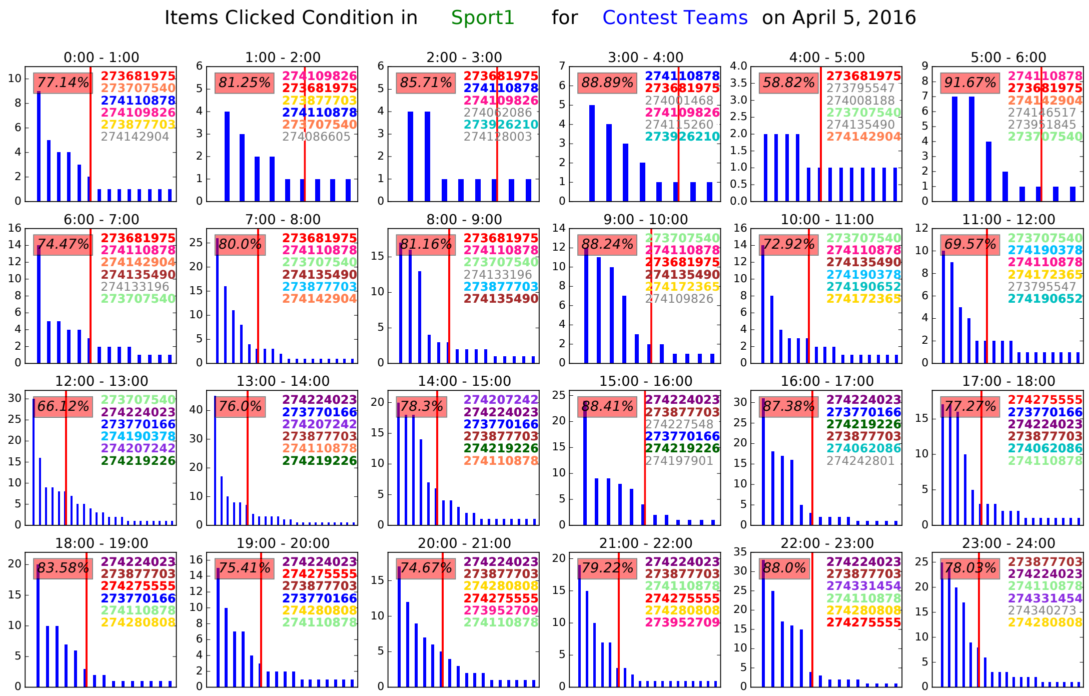


Figure 4.2: Top clicks condition of *contest teams* on *Sport1* on April 5, 2016. Frequency distributions of top clicked articles are visualized hourly. For the top 6 clicked articles, if an article is ranked in top 6 across two neighbored hours, it's painted with the same color in the both neighbored bins.

4.2.2.1 Clicks Pattern Analysis for *Sport1*

First, we draw the histogram of clicks regarding recommendation requests on items in portal *Sport1*. Considering that PLISTA only shows a part of the computed recommendations to the participants, we assume that the click patterns differ amid contest teams scope and the whole PLISTA scope. Thus the figures are also divided into cases for the two scopes specifically.

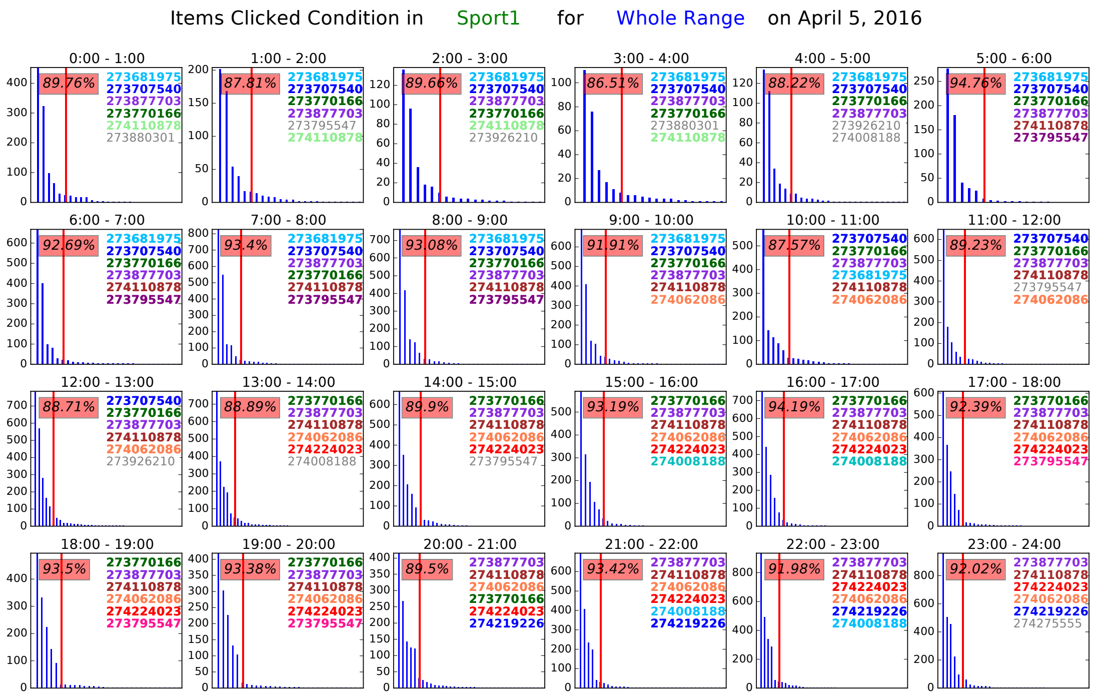


Figure 4.3: Top clicks condition of *whole* range on *Sport1* on April 5, 2016. Frequency distributions of top clicked articles have been visualized hourly. For the top 6 clicked articles, if an article is ranked in top 6 across two neighbored hours, it's painted with the same color in the both neighbored bins.

Figure 4.2 shows the clicks distribution among contest teams. In order to track the top clicked items in a time sequence, we draw a figure for each hour on April 5, 2016, and generate 24 sub-figures covering the whole day. In each sub-figure, news items are located on the x-axis as points sorted by the click frequency in descending order. A red vertical line separates the 6 most frequently clicked items (6 is the required number for the recommendation slots in NewsREEL). For most 1 hour intervals, we observe the power law distribution. The percentage of clicks occupied by the top six items is shown in the red boxes in the sub-figures.

To analyze how popularity transitions into the future, we highlight the item_ids of the six most popular items in the top right corner of each sub-figure. We color item_ids to track individual items throughout the plot. Items tinted in gray only appear in a single one hour interval. From Figure 4.2, we see that on April 5, 2016, items ranked in the top three manifest more continuity, i.e. they are more likely to re-appear in the next hour's top clicked 6 items group.

Aside from the scope of contest teams in ORP, we also find the power law distribution of clicked items in the whole PLISTA range. In Figure 4.3, along with the increasing number of distinct clicked items in the "whole" range, the power law distribution of clicked items is even more significant. In all one hour time windows, more than 87% clicks are comprised of top 6 items. The distribution can be described by Zipf's law. The significantly high occupation of the six most frequently clicked items show the hint on paying more attention to the short head with higher business value.

4.2.2.2 Clicks Pattern Analysis for *Tagesspiegel*

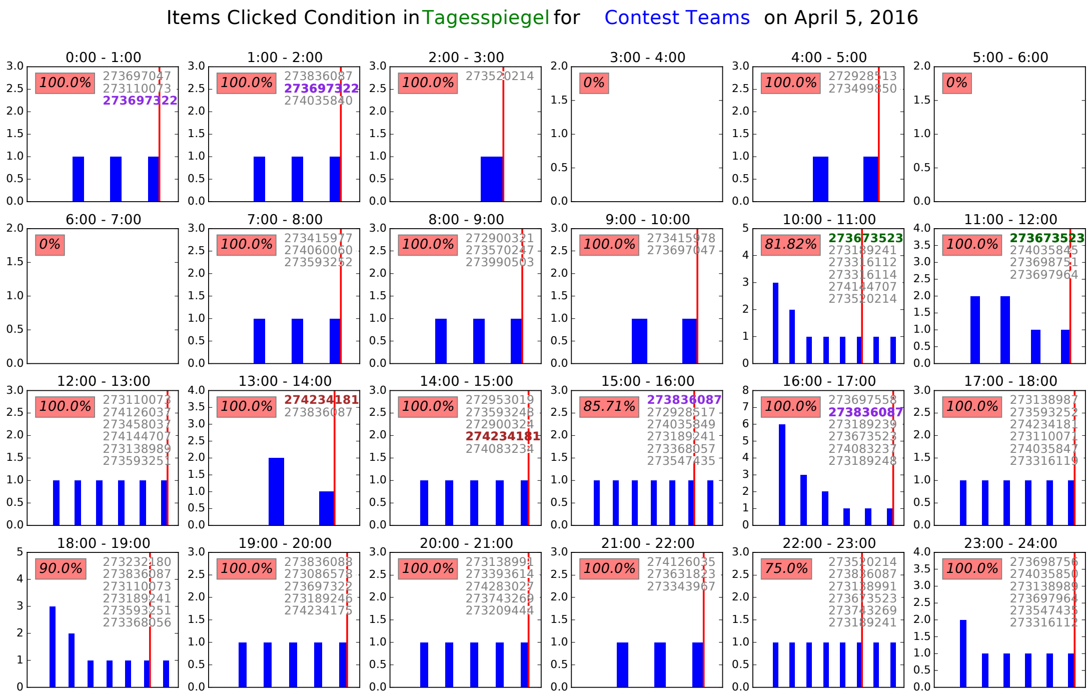


Figure 4.4: Top clicks condition of *contest teams* range on *Tagesspiegel* on April 5, 2016. Frequency distributions of top clicked articles have been visualized hourly. For the top 6 clicked articles, if an article is ranked in top 6 across two neighbored hours, it's painted with the same color in the both neighbored bins.

All data analysis in the subsections above was on data collected for the portal *Sport1*. Yet as the second largest portal in the contest of the year 2016, the clicks analysis of *Tagesspiegel* is also analyzed. We redraw the histogram of the clicked items for *Tagesspiegel* as what we did for the portal *Sport1*. As Fig 4.4 shows, there are so few clicked items in the range "contest teams". Some one hour intervals have less than six clicks in total. Thus no obvious regular pattern can be found in this range. Even when considering the whole PLISTA range of clicks, Figure 4.5 shows a flatter power law distribution. The top items only account for 20–35% of clicks. In addition, we observe more variation in the most frequently clicked items such that many items appear only for a single hour in the *top 6* group. *Sport1* exclusively provide news with topic on sports. While *Tagesspiegel* covers a wide range of topics including politics, economy, sports, and local news. We therefore hypothesize that the flattened bias is caused by the higher diversity in topics.

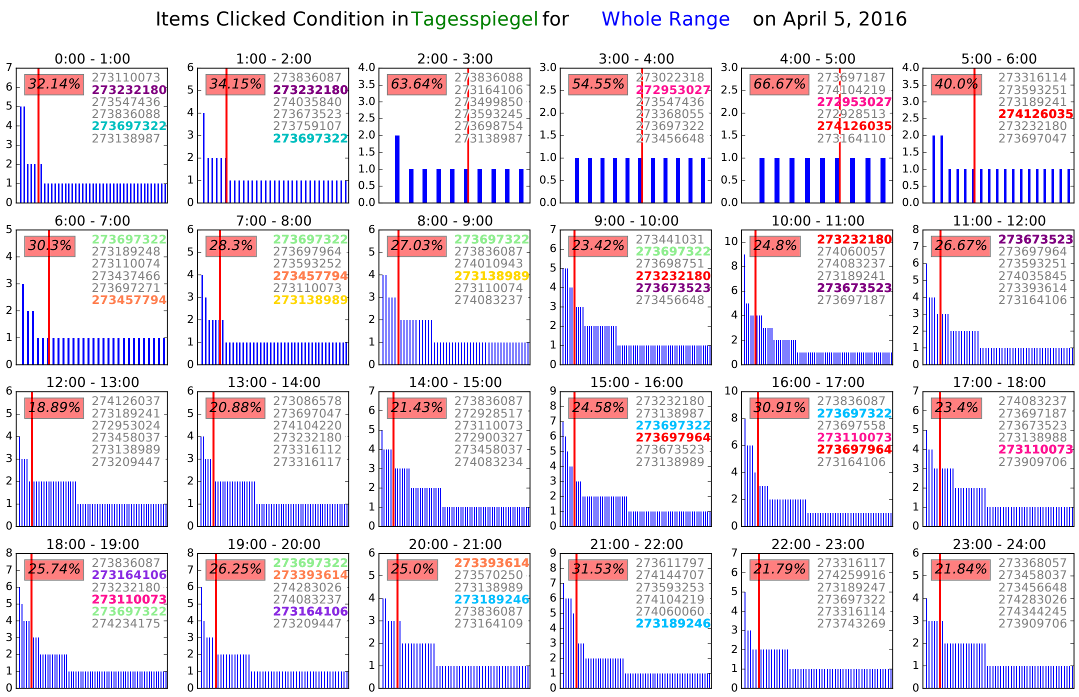


Figure 4.5: Top clicks condition of *whole* range on *Tagesspiegel* on April 5, 2016. Frequency distributions of top clicked articles have been visualized hourly. For the top 6 clicked articles, if an article is ranked in top 6 across two neighbored hours, it's painted with the same color in the both neighbored bins.

4.2.3 Predicting Power on *Impressions* and *Clicks*

Having observed the good recommendation effect of directly dealing with the popularity of *Impressions* and *Clicks* in the online challenge, we study the impact of impressions and clicks from a temporal perspective. We use the Jaccard Similarity for tracking the trends.

4.2.3.1 Jaccard Similarity as Representation of Correlation

In this subsection, we quantify the continuity of most frequently clicked items and analyze this continuity behavior concerning contextual factors such as time of day and day of week. Jaccard Similarity, as defined in Equation 4.1, is a metric to measure the similarity of two sets A and B. The value of this metric equates to the cardinality of the intersection divided by the size of union of these two sets. In our scenario, A and B refer to the sets of the six most frequently clicked items of two neighboring one hour time slots. The higher the Jaccard similarity, the more significant of the biased concern proved in this neighbored time bins.

$$\text{Jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{4.1}$$

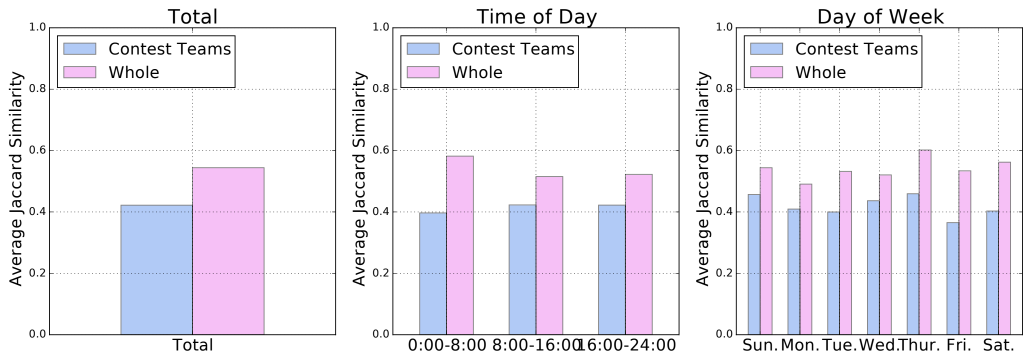


Figure 4.6: Jaccard similarity of top clicked items between continuous hours for *Sport1* in the week 3–9 April, 2016. The cyan bar represents the condition of the contest teams, the pink bars represents the whole scenarios. We analyze the Jaccard Similarity from the perspective of total, time of day, and day of week specifically.

We expand our view from a single day to the week 3–9 April, 2016, and obtain $24 \times 7 = 168$ one hour time windows. Therefrom, we derive 167 pairs of subsequent time windows to compute the average Jaccard similarity. Figure 4.6 illustrates our findings overall, for specific times of day, and for each weekday. We distinguish the contest scope and the whole PLISTA scope by cyan and pink colors. Throughout the three subfigures, we notice that the Jaccard metric on PLISTA whole scope exceeds its value on the contest scope. The gap is most obvious in the night (0:00–8:00). Independent of context, Jaccard scores fall in the range of 40–60%. This signals that more than half of the most popular items re-occur in the next hour's *top 6* group. Thus, recommending popular items guarantees a good chance to perform well. This explains the good performance of the "baseline" in previous editions of NewsREEL.

4.2.3.2 Predicting Ability Comparison *Impressions* v.s *Clicks*

Empirically speaking, "Most Viewed" approach always performs well w.r.t. CTR. This is the motovation for us to study the impact of impression and clicks for computing recommendation for the next hour in detail.

Based on the detected click patterns (discussed in Subsction 4.2.2.1 for the portal *Sport1*), we compare recommender strategies based on impressions and strategies based on clicks. Since the six most frequently clicked items receive more than 80% of all hourly clicks, correctly capturing the *top6* group in a time bin is quite adequate for providing a reasonable recommendation. We define predicting ability here by the Jaccard similarity between the set of recommended items and the set of the six most frequently clicked items in a specific hour. The higher the Jaccard similarity, the stronger the recommending set representing the predicting ability. The six items most frequently viewed in the *last hour* form the recommending set "Most Viewed". In addition, the six items most frequently clicked after having been suggested in the *last hour* characterize the recommending set "Most Clicked".

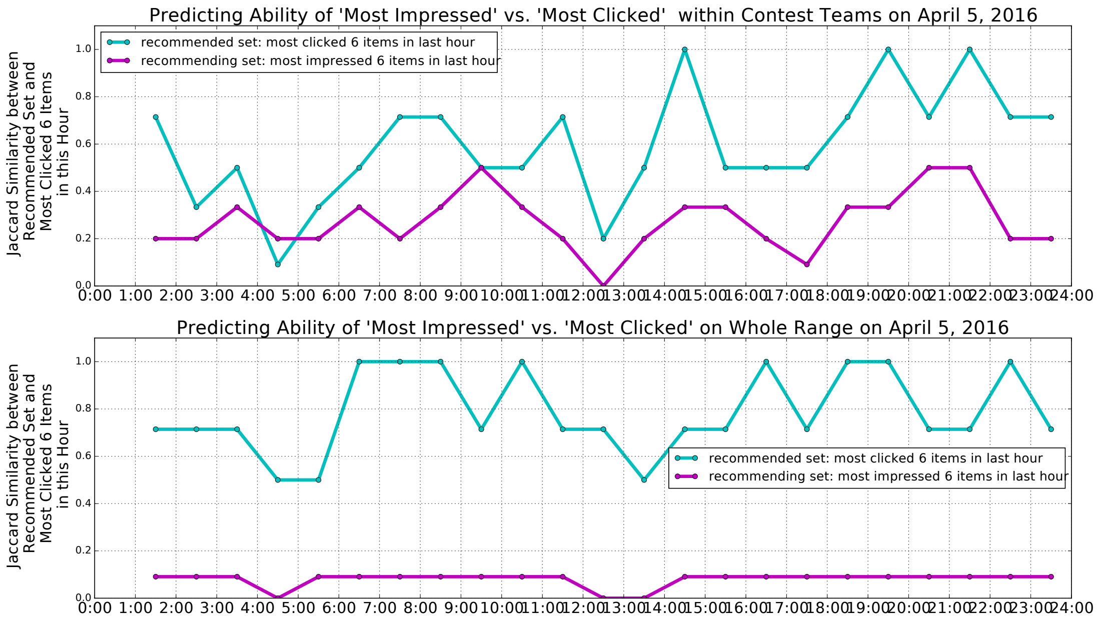


Figure 4.7: "Most Viewed" vs. "Most Clicked" on predicting next Hour "Most Clicked" in *Sport1* on April 5, 2016.

Figure 4.7 shows both methods' performances over time. The cyan curve refers to the predicting ability of recommending set "Most Clicked" while the magenta line refers to the predicting ability of recommending set "Most Viewed". The upper subfigure shows the comparison of "Most Viewed" and "Most Clicked" in the range "contest teams", and the bottom subfigure presents the same comparison in range "whole PLISTA". In both "contest teams" range and "whole Plista" range, *most clicked* 6 items in last hour always show a higher Jaccard Coefficient with the *most clicked* 6 items in this hour than the *most viewed* 6 items in last hour do. Evidently, "Most Clicked" outperforms "Most Viewed" in both scopes. This indicates that at least on 5 April 2016, users' reactions to recommendations let the system better predict future clicks than what they read. The outperformance indicates that clicks on recommendation requests as a time series data has its self-predicting ability and deserves more attention even compared with impressions.

4.2.4 Discussion

In this section we have discussed the algorithms used in the news recommendation challenge. Given the observations that popularity based approaches always represent a better recommendation effect, we have analyzed the pattern of *clicks* and *impressions* in dominant portals *Sport1* and *Tagesspiegel* from temporal view. Typically in news portal *Sport1*, the number of *clicks* of individual articles shows a significant power law distribution on an hour basis. The duration of the dominant role of an individual article can be lasting for days, which is much longer than we expected. To solve **RP 4.1: To what extent is the predicting power of the popularity bias for the CTR of the news recommender?**, we use the Jaccard similarity between the predicted ranking list and the real clicked list to represent the prediction ability. The defined measurement reflects that the *clicks* set gains more prediction power on the clicked items in the next hour than the *impressions* set does.

4.3 Parsing and Mining Topic Concern Bias in News Stream

Users' reading behavior on digital news tends to be biased according to existing studies analyzing the news stream consumption [18]. Two insightful observations have been acquired from our participation in NewsREEL2016. First, the duration of a news article being popular can last longer than we expected. Second, the popular articles can be related to similar topics. To study the content-based similarity between news articles, we spread the observation and analysis duration on the news stream provided from NewsREEL2016 to 8 months. We focus on the text level of the news items and study the content bias in addition to the *click* and *impression* user behavior.

To express such topic concern bias, we introduce two concepts regarding terms in news articles – Term Concern Degree (TCD) and Topic Ranking Bias (TRB). TCD is defined by term's document frequency weighted on impressions or clicks, such that the importance of a specific term can be calculated from both *news text stream* and *user behavior* aspects. We apply ranking similarity Average Overlap (AO) between two neighbored terms lists to more precisely denote topic consistency (or content topic bias) and name it as TRB.

We analyze how users interact with news items and recommendations provided in the NewsREEL challenge 2016. We distinguish two kinds of users' behavior: *impressions* and *clicks*. *Impression* denotes users' direct viewing through the main page or site navigation, while *click* represents users' clicks on the recommended items at the bottom of articles.

Research questions we are studying in this section include: 1) temporal topic concern bias via continuous dates regarding a specific user behavior type; 2) content consuming consistency between different user behavior types.

4.3.1 Content Parsing for Popular Items

Analyzing the item distribution for the portal *Sport1*, we find a power law distribution. We study the concrete content of the most frequently clicked news items. Table 4.2 illustrates the four items occurring most frequently in the *top 6* group. It lists their item_id, the duration contained in the *top 6*, their ranking trends, and the date the items had been created. Regarding the ranking of these 4 items, we find that even within the shortly analyzed timeframe, they stay in *top6*; but their concrete rank changes. Tracking or predicting such fluctuation is also one of our focus attentions in the following sections. In addition, the duration of the items being popular subverts our expectation. Even within the tracking day, April 5, 2016, as the table shows, all the four items remained in the *top 6* for at least eleven hours, though we had expected considerably less time as news continuously emerge. When observing the creating dates of these items, it even indicates a noticeably longer life-cycle of news than we anticipated. Taking a typical example for item *273681975* with headline "Guardiola macht Götze froh": it was created on April 2, 2016, and dominated the *top 6* news group three days later on April 5, 2016.

Table 4.2: Stable top clicked items condition on April 5, 2016.

Item Id	Being in Top6	Ranking	Created At
273681975	0:00–11:00	Most of the time 1st	2nd, Apr. 2016
273707540	0:00–13:00	2nd $\rightarrow$ 1st	2nd, Apr. 2016
273877703	0:00–24:00	3rd/4th $\rightarrow$ 2nd $\rightarrow$ 1st	3rd, Apr. 2016
273770166	0:00–21:00	2nd $\rightarrow$ 3rd/4th $\rightarrow$ 1st	2nd, Apr. 2016

Next, we analyze the categories, titles, and descriptions of these four items in order to get a better understanding of the contents. Table 4.3 highlights the co-occurring terms in these four items. Among those, we see that the breaking news that coach *Pep Guadiola* will leave *Bayern* attracted many users' interest on relevant articles. This shows, at least in the domain of *Sport*, the articles mentioning the most breaking events attract the most traffic, and these breaking events are the root cause of the popularity of articles. Considering this point, though former participants of NewsREEL mentioned that pure content-based recommenders frequently suggest articles with minor clicks, we would like to supplement that for popular items, content similarity does affect the effect of recommenders.

Table 4.3: Stable top clicked items description on April 5, 2016.

Item Id	273681975	273707540	273877703	273770166
category	fussball	intenational-fussball	fussball	fussball
title	**Guardiola** macht **Götze** froh	**Van Gaal** watscht di Maria ab	Robben:"**Van Gaal** ist wie **Guardiola**"	Mittelfeldbestie **Götze**: Wechsel nur im Notfall.
text	Der Coach des **FC Bayern** lässt Youngster Felix **Götze** erstmals mit den Profis trainieren. Javi Martinez und Manuel Neuer stehen derweil vor dem Comeback.	Als Rekordtransfer geholt, nach nur einer Saison wieder vom Hof gejagt. Jetzt geht die Geschichte zwischen United-Trainer Louis **van Gaal** und Angel di Maria in die Verlängerung.	Arjen Robben vergleicht den ehemaligen **Bayern**-Coach Louis **van Gaal** mit dem aktuellen Trainer Pep **Guardiola**. Mit dem **FC Bayern** will Robben noch viel erreichen.	Mario **Götzes** beherzter Auftritt gegen Frankfurt zeigt: Er will sich unbedingt beim **FC Bayern** durchsetzen. Der Verein mauert noch beim Thema Transfer.

4.3.2 Topic Concern Bias Definitions

To understand the topics embedded in the news streams, researchers define topics by different approaches. Normally, terms or named entities within news text are treated as a topic directly [140–142, 154]. In this case, grouped counting on term frequency or document frequency are represented as the hotness or concern degree of a topic. Nevertheless, for digital news publishers, users' consuming behavior (e.g. *impressions* or *clicks*) is also an important auxiliary indicator for their topic hotness in addition to the pure news text stream. Hotness metric which incorporates the textual terms and user behavior is then our main objective in this subsection.

We define the hotness score of a term within a time window as Term Concern Degree (TCD), which is calculated on the basis of terms' document frequency weighted on specific users' behavior. The higher the score for a term, the hotter the topic content is consumed. On the top of TCD, we rank the terms as a list within time window. The ranking similarity (Average Overlap) between two lists is used to represent the degree of consistence for the topics ranking, i.e. the bias of content topic consumed by users. Through the visualized topic bias in four of the analyzed news portals, we found that the reading behavior of the audiance of different news portals varies w.r.t the topic selection bias.

4.3.2.1 Definition of Term Concern Degree

We define the Term Concern Degree (TCD) for terms on a daily basis, the procedure of collecting the candidate terms goes through the following steps:

a) pre-processing of article items: it includes removing garbles, stemming and removing stop words;

b) ranking article items: according to the number of impressions or number of clicks on a specific date, the articles are ranked with top K (K=20 in the experiment setting);

c) normalizing the impressions count: for the top K articles, the normalization is realized based on their weights w.r.t. the concrete user behavior type;

d) sorting the topic concern terms: according to the calculated TCD, the top terms are sorted and form the sorted term list.

$$TCD(t,d) = w_{title} * score_{t,d}^{title} + w_{text} * score_{t,d}^{text} \tag{4.2}$$

$$norm_d = \sum_{i \in D_{d,K}} \#impression_{i,d} \tag{4.3}$$

$$score_{t,d}^{title} = \frac{\sum_{i \in D_{t,d}^{title}} \#impressions_{i,d}}{norm_d} \tag{4.4}$$

$$score_{t,d}^{text} = \frac{\sum_{i \in D_{t,d}^{text}} \#impressions_{i,d}}{norm_d} \tag{4.5}$$

Notation $TCD(t,d)$ illustrates the concern degree of term t on date d. Here we introduce the procedure for calculating $TCD(t,d)$, and take the *impressions* behavior as an example. Eq. 4.2 to Eq. 4.5 list the main calculation steps for TCD. From the step b) ranking article items in the last paragraph, we get the top K candidate articles set for the date d as set $D_{d,K}$. As Eq. 4.3 shows, the normalization factor $norm_d$ across set $D_{d,K}$ (the set comprises of top K articles at date d concerning their *impressions*) is the summation of the number of impressions for all articles in the set $D_{d,K}$. For each article in the candidate set, there are two textual fields: title and text. Terms appeared in these two fields are assigned with importance weight w_{title} and w_{text} respectively (w_{title} is set as 0.6 while w_{text} as 0.4 in the current experiment setting). As Eq. 4.2 explains, $TCD(t,d)$ is the aggregated weighted score of $score_{t,d}^{title}$ and $score_{t,d}^{text}$ in consideration of w_{title} and w_{text}. And the definitions of $score_{t,d}^{title}$ and $score_{t,d}^{text}$ are similar as Eq. 4.4 and Eq. 4.5. $D_{t,d}^{title}$ and $D_{t,d}^{text}$ are the two subsets of $D_{d,K}$. $D_{t,d}^{title}$ represents the articles, whose titles contains term t on date d. While $D_{t,d}^{text}$ includes the articles, whose text description contains term t. As Eq. 4.4 shows, for the *title* field, the number of article impressions in the set $D_{t,d}^{title}$ divided by the normalization factor $norm_d$ is defined as the topic concerning score regarding term t on *title* field on date d for users' *impressions* behavior. As the same token, we can also produce the topic concern score on text field as Eq. 4.5 shows. Being aggregated on title and text fields, the general score $TCD(t, d)$ is used to sort terms based on users' *impression* behavior. Similarly, such TCD can also be calculated for user behavior *clicks*.

The TCD is defined as the combination of text description and user behavior, we also argue that it is a special existence of Document Frequency (DF). For instance, when talking about the *impression* behavior, the number of impressed articles can be seen as the number of repeated documents detected in the corpus. Therefore, TCD is essentially a transformation of DF with the advantage of fields fusion.

4.3.2.2 Topic Ranking Bias (TRB) based on TCD

Having ranked the terms according to their TCD within each time bin, we gain an intuitive view of the topics ranking representations. By tracking the *ranking similarity* of the topic ranking lists in neighbored time bins continuously, we could see the degree of bias of the topics ranking along the time axis. This *ranking similarity* is defined as TRB in this section.

The most prevalent scenario of applying *ranking similarity* is the Web Search Engine (WSE), where lists of ranked documents computed for the same search query from different search engines are compared [155]. The ranking similarity describes the degree of similarity between the analyzed search engines. Since only the top ranked documents in search engines result list are relevant to most users, classical ranking similarity metrics, such as Kendall's Tau are not so applicable in this scenario. Thence, measurements like AO and Ranked Bias Overlap (RBO) are proposed to deal with these issues and have proven to be working well [156]. In our scenario, for a ranked term list, since the terms with higher scores gain more importance, it also fits the *top-weighted* condition. Thence AO and RBO can be used for measuring the list bias condition. Yet RBO is based on the convergent series of weights in *indefinite* lists, it's not suitable in our *definite* top K terms ranking condition. Thus AO is chosen as the indicator of topics ranking consistency of *definite* lists.

Given two non-conjoint and weighted top ranked lists S and T, at depth K, the size of intersection of two ranked lists over K, say $|S \cap T|/K$, denotes the similarity or the overlap between two lists at depth K. For each depth $k \in \{1 \ldots K\}$, we calculate the $overlap@k$. The average of the cumulative overlaps at these increasing depths derives to the value $AO@K = \sum_{k\in\{1...K\}} overlap@k/K$ [156, 157]. Since $AO@K$ is calculated for paired terms' ranking lists, which essentially represents the content topic consuming persistence, we call it Topic Ranking Bias (TRB) in this chapter.

4.3.3 Discussion

In this section, our analysis targets on the research problem **RP 4.2: How to parse and analyze the topic concern content for the popularity bias in the news consuming stream?** First, the content level representation of the popular articles has been analyzed in digital news portals. For some frequently co-occurred hot articles, the extracted topics explain the reason leading to the popularity. In order to represent the concern degree of a specific topic, we define TCD as the weighted document frequency of the designated term over all the popular articles. Nevertheless, TCD only records the topic concern degree or bias of a single topic. To represent the ranking bias of the topics between two time windows or two types of user behaviors, the AO is applied as the measurement for ranking similarity.

4.4 Topic Ranking Analysis: Temporal or Behavioral View

In the last section, we proposed use AO to represent the topic-level ranking bias between two ranking lists. The views of the topical list-wise ranking bias can be either *temporal* or *behavioral*. In this section, TRB-t and TRB-b are defined as ranking bias measurements for these two views specifically. For TRB-t, we determine the type of user behavior and extract the pair of ranked lists on temporally neighbored time bins to represent the temporal topic concern bias degree. The higher TRB-t, the higher consistency of the content users have consumed along the temporal evolving axis. For TRB-b, on the contrary, we fix the date and extract the terms ranked lists regarding two different user behaviors and calculate the ranking bias degree. TRB-b denotes the reading content consistency between behavior *clicks* and *impression*. It also indicates the suitable moment for using hot topics discovered in the hotly impressed articles to give the recommendation on the *clicks*. To analyze the biased ranking from these two views, we conduct our analysis for the ranking bias based on the data collected for the four news portals caught in the challenge.

4.4.1 Temporal View of Topic Concern Bias

In this subsection, we view the topic concern ranking bias from the temporal perspective. We calculate the TRB-t over N days on a daily basis for the four candidate German news publishers within the traced time period. We extract several ranked term lists to zoom in and observe some typical topic content variation period. The statistics of the neighbored TRB-t for different publishers are shown in Tab. 4.4 (TRB-t w.r.t. *impressions*) and

Tab. 4.5 (TRB-t w.r.t. *clicks*). The tables list the length of day sequence, mean value and standard deviation of TRB-t. In addition, peaks ratio (peak is defined as where TRB-t value greater than 0.6) and valleys ratio (valley defined as where TRB-t value less than 0.2) over all temporally neighbored pairs on the date axis are also provided.

Table 4.4: Statistical result of TRB-t regarding *impression*.

	# neighbors	mean	std	Peaks Ratio(%)	Valleys Ratio(%)
KÖLNER STADT ANZEIGER	72	0.343	0.175	9.72	27.78
GULLI	278	0.517	0.199	34.53	11.15
TAGESSPIEGEL	79	0.245	0.161	3.80	41.77
SPORT1	134	0.344	0.072	0.75	1.49

Regarding user behavior *impression*, Tab. 4.4 shows the ranking bias for the four news portals. The degree of the ranking bias demonstrates differently for specific news portals. The news portal GULLI shows a generally higher TRB-t, with the mean value 0.517 and peaks ratio 34.53% over the whole 278 neighbor dates pairs. It indicates that in publisher GULLI, users' overall impressing behavior remains high content level consistency along the time axis. On the other side, the news portal TAGESSPIEGEL has a low mean value of TRB-t 0.245 and the dominant valleys ratio 41.77%. They imply that temporal topic bias is not so obvious in the TAGESSPIEGEL.

Table 4.5: Statistical result of TRB-t regarding *click*.

	# neighbors	mean	std	Peaks Ratio(%)	Valleys Ratio(%)
KÖLNER STADT ANZEIGER	45	0.356	0.243	13.33	26.67
GULLI	250	0.364	0.226	11.60	27.20
TAGESSPIEGEL	55	0.325	0.130	1.82	12.73
SPORT1	117	0.337	0.107	0.85	11.11

Analogously, the statistics of TRB-t for the four publishers on the *click* behavior are shown in Table 4.5. The mean values of TRB-t vary within a small range, 0.325 to 0.364. The comparatively high standard deviation (greater than 0.2 in KÖLNER STADT ANZEIGER and GULLI) denotes that temporal consistency of topic concern content of the *click* behavior is not stable but specific period interval dependent. Generally, from the temporal view, the *click* behavior shows weaker topic ranking bias as *impression* does.

For intuitively representing the topic concern bias variation, we choose some example periods and compare the overlapping content of the ranked term list separately. The first example period we extract is from 10th July 2016 to 23rd July 2016 of publisher GULLI. The terms ranking lists for each date for *impression* are shown in Fig. 4.8 and *click* behavior are shown in Fig. 4.9. Within these figures, the red blocks stand for the continuous terms appearing between two neighbor dates. The higher number of red boxes, the higher the content similarity between the neighbored time windows. In Fig. 4.8, TRB-t on *impressions* exhibit a convex peak in the observed period. Analyzing the content, we can see that the topic changed quickly in the first four days, popular topics like **Windows Operation System updating** , **Pokeman Go** and **Youtube favorite PewDiePie** haunt from time to time across the time windows, while **NASA released report on the Second Moon** suddenly attract most attention from 15th July 2016 to 21st July 2016. Obviously, **Second Moon** caused users' heavy reading bias because of their worries about **Asteroid Collision on Earth**. In *clicks* term lists in Fig. 4.9, the attention on **Second Moon** also shows strong biased ranking effect since 16th July 2016, yet with a shorter lag compared with *impression*. Such peak period showing up in the TRB-t sequence indicates that the bias moment arriving and hints for caching the valuable biased terms for making recommendations in the near future.

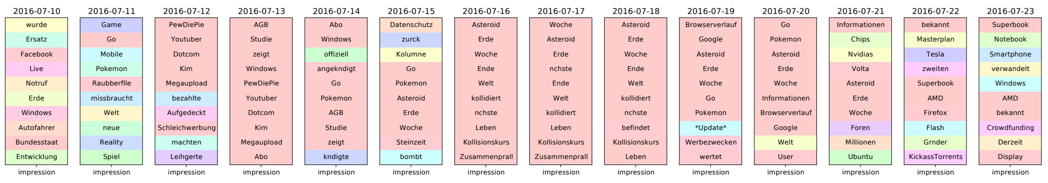


Figure 4.8: Terms ranked list regarding users' impression behavior for the publisher Gulli (www.gulli.com).

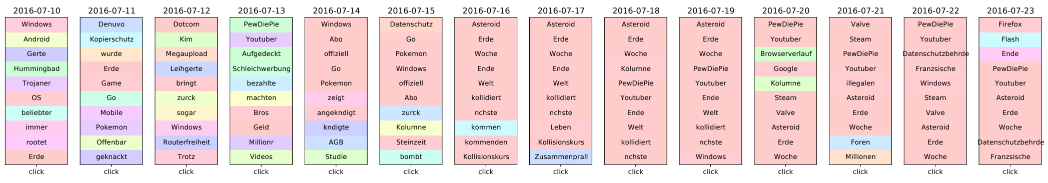


Figure 4.9: Terms ranked list regarding users' click behavior for the publisher Gulli (www.gulli.com).

The second example period is from 25th January to 29th January 2016 for publisher SPORT1. The interesting phenomenon can be found within this period is that TRB-t trends of *impression* and *click* go into opposite directions. Seen from the content in these terms lists, transfer market transaction stably attracted attention for *impressions*. While the *clicks* on recommendation varies rapidly between dates. In this period, temporal topic concern bias shows more prediction power for *impression* behavior than *click*.

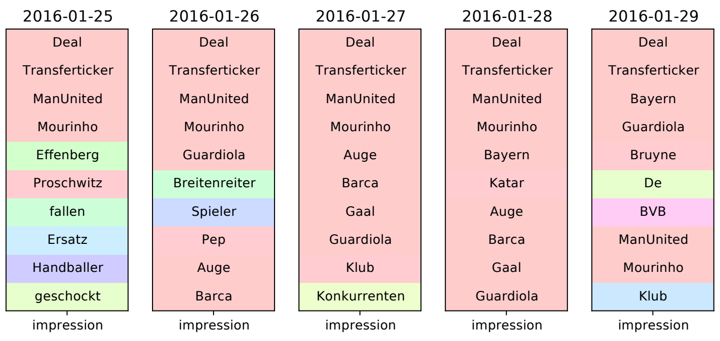


Figure 4.10: Terms ranked list regarding users' impression behavior for the publisher Sport1 (m.sport1.de).

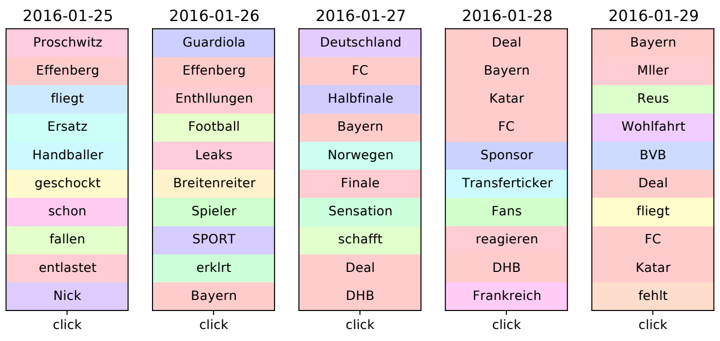


Figure 4.11: Terms ranked list regarding users' clicks behavior for the publisher Sport1 (m.sport1.de).

4.4.2 Behavioral View of Topic Concern Bias

Though TRB-t is calculated along the time axis to record the continuous ranking bias degree for a specific user behavior, no mutual relation between different user behaviors can be detected in this metric. In this section, we use metric TRB-b to observe users' topic concern bias from a behavioral view, typically for *impressions* and *click*. To this end, we extract the terms ranking lists generated by score on top of *impression* and *click* within the same time window. The ranking similarity between the two lists are then calculated, this similarity stands for the degree of topical bias discovered between *impression* and *click* within this time window.

Based on the example time period we observed for TRB-b, Tab. 4.6 represents the concrete statistical result among different publishers. The mean value of TRB-b for TAGESSPIEGEL and SPORT1 can reach 0.667 and 0.585 respectively. The peaks ratio of TAGESSPIEGEL 88.71% signals that for the most time in this publisher, content topics users focus are similar across the *impression* and *click*. Or from another perspective, in publisher SPORT1, the low standard deviation 0.074 also reflect the ubiquitous con-

	# neighbors	mean	std	Peaks Ratio(%)	Valleys Ratio(%)
KÖLNER STADT ANZEIGER	51	0.383	0.283	23.53	33.33
GULLI	254	0.469	0.237	38.58	20.08
TAGESSPIEGEL	62	0.667	0.216	88.71	9.68
SPORT1	118	0.585	0.074	44.92	0.00

Table 4.6: Statistical result of TRB-b on the four publishers.

tent consuming overlapping effect between *impression* and *click*. But in KÖLNER STADT ANZEIGER and GULLI, the low mean value and the high variance signal the quick changes of TRB-b. It shows that the applicability for making recommendations on *clicks* from topic concern in *impressions* is only periodically valid.

Analogously to the analysis based on TRB-t, we retrieve some periods from the tracked time span to see the overlapping effect of terms concern lists between *impressions* and *clicks*. We visualize the ranked term lists slightly different from terms list for TRB-t. Within each time window, term lists ranked by scoring schema TCD regarding *impression* and *click* are grouped together, the common terms appeared in the paired lists are all colored in red. Thus the larger the red area in the grouped pair list, the more topic concern bias exists between *impression* and *click*. Within these time windows, biased viewed content of *impression* possesses the ability of foretelling hot content in the clicked recommended items. Within each time window, the value of Average Overlap (AO) between the two lists and the number of overlapped documents are also provided.

The first sample interval retrieved is from 3rd Feb. 2016 to 8th Feb. 2016 in publisher KÖLNER STADT ANZEIGER. As Fig. 4.12 shows, accompanied by the arrival of **Rosenmontag**, which is celebrated in Rhenish carnival, users' collective attention has been drawn on the events in both *impression* and *click* in Cologne area. Though on 7th February, some sub topics about the carnival diverse the biased consistency between *impressions* and *click* a little bit, they are united on 8th February again because of the coming of the exact **Rosenmontag** on this day. And the interaction of articles also reaches the highest – 14 on this day.

The second period is from 26th Aug. 2016 to 2nd Sept. 2016 in publisher GULLI as presented in Fig. 4.13. During the first four days, the topic concerns between *impression* and *click* are quite different, almost no behavioral bias can be detected. For instance, on 27th August 2016, though **Windows** and **Intel** gained more concern in users' *impressions*, **Pokemon Go** and **Russian Spy** were the favorable content when users' clicked on recommendations. However, when the astronomical topic **SETI** arrived on 30th August 2016, *impression* and *click* together become highly biased towards it. Such cross behavioral bias lasts for several days.

4.4.3 Discussion

With topic ranking bias (TRB) defined on top of topic concern bias, we represent the bias degree based on the ranked list level. The comparison analysis has been set up from two perspectives: temporal view and user behavioral view. Thereby we solve the **RP 4.3: How could we understand the topic ranking bias from temporal view and user behavioral view?**

The temporal view of the ranked list bias is reflected by TRB-t, an indicator tracking the overlap ratio between the neighbored time windows. The temporal view analysis on such bias gives the hint on the predicting power with the influence of temporal factors into consideration. From the analysis on the continuous time windows, publishers *Gulli* gains most temporal topic concern bias regarding *impression* behavior.

From the behavioral view, the biased topic concern overlap between different user behaviors are calculated as TRB-b. *impression* and *click* are the two source behaviors contribute to the value of TRB-b. The behavioral ranking bias analysis shows the predicting power of one user behavior (like impression) on the other (like clicks). For publisher *Tagesspiegel*, the behavioral bias represent more obvious characteristics.

4.5 Conclusions

In this chapter, we went through our participation in the NewsREEL news recommendation challenge in the year of 2016. The outperformance of the popularity-based approaches attract our attention. Thus we analyze the popularity bias and its explanation in the news recommenders from multi-views.

On the *article level*, the popularity is tracked on the hour basis across a day for behavior *impression* and *click* respectively. By applying Jaccard similarity of the popular articles lists between the neighbored time windows, the consistency of article popularity was represented. This list-wise popularity temporal consistency indicates the predicting power of a designated behavior. It turned out that article level bias on behavior *click* possesses higher predicting power than *impression* does.

Topic level understanding of the popularity shows the explanation of the bias in the trend. TCD is defined based on the weighted terms document frequency. It represents the popularity of a topic within a time period. For the topic ranked list based on TCD, TRB uses AO to quantify the ranking bias for the compared lists. Ranking list based on TRB can be analyzed from either temporal view or behavioral view. From the temporal view, TRB value is generally high in the *impression* behavior than the *click* behavior. While from behavioral view, the predicting power of *impression* on *click* is captured in the majority portion for both publishers *Tagesspiegel* and *Sport1*. Content level topic concern bias understanding is thus helpful for the behavioral mutual prediction.

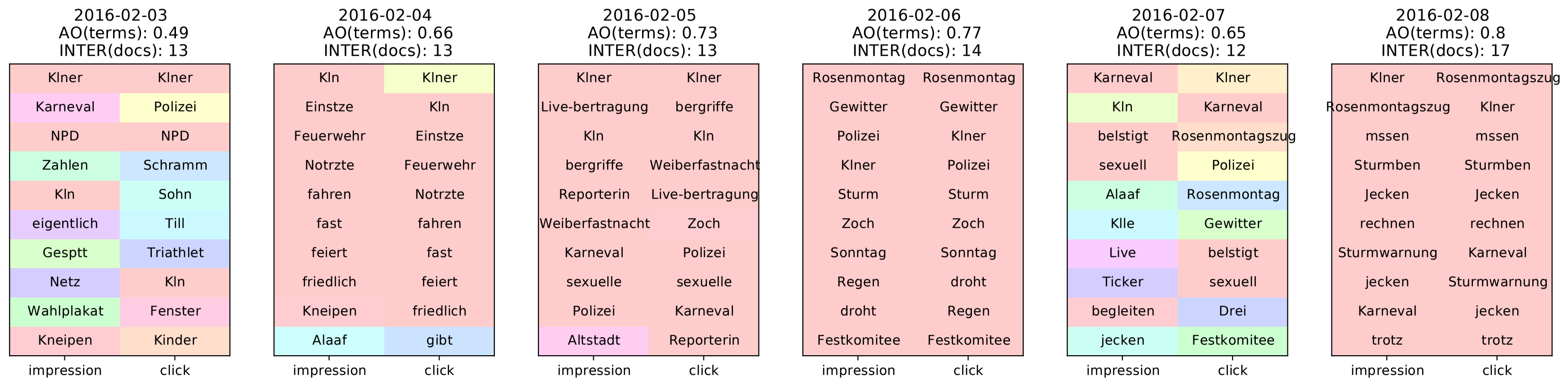


Figure 4.12: Terms ranked lists comparison on impressions and clicks for the publisher KÖLNER STADT ANZEIGER (www.ksta.de).

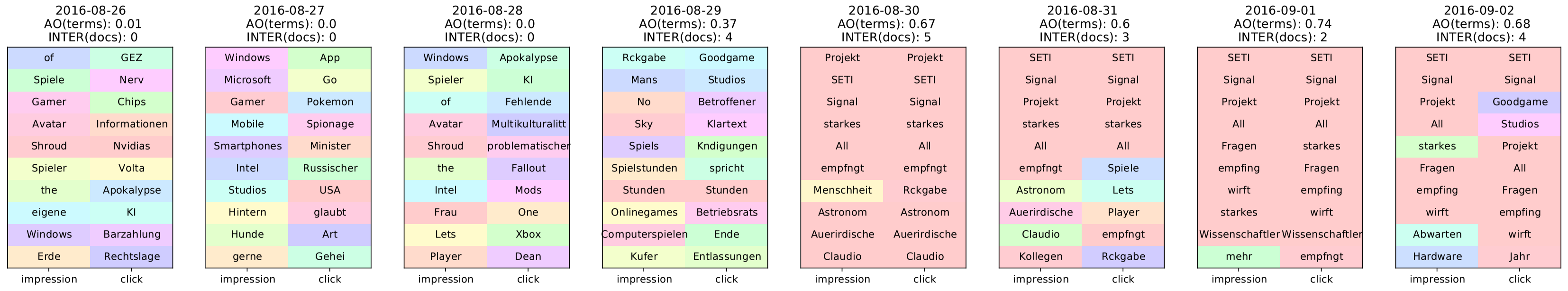


Figure 4.13: Terms ranked lists comparison on impressions and clicks for the publisher Gulli (www.gulli.com).

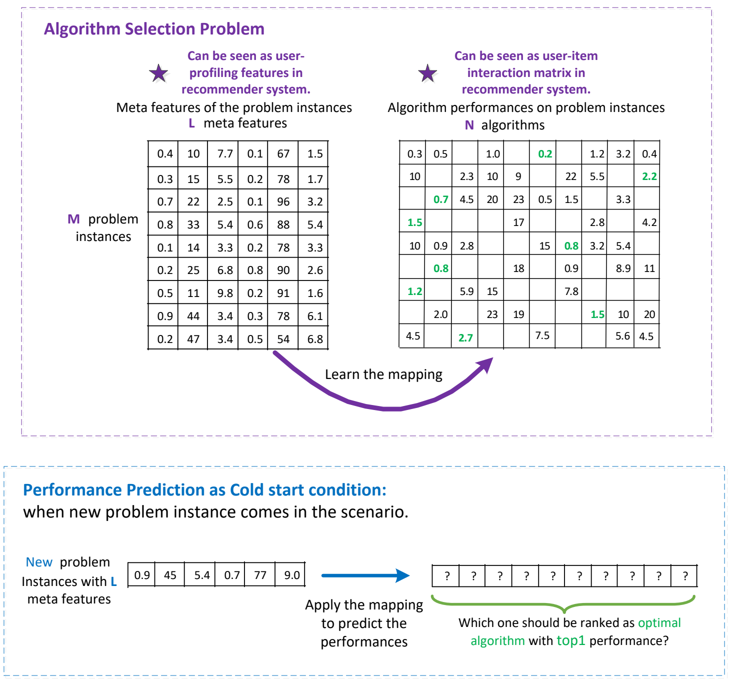
CHAPTER 5

ALGORITHM RECOMMENDATION: EVALUATION BIAS TRADE OFF

Algorithm Selection (AS) tasks are dedicated to find the optimal algorithm for an unseen problem instance. With the knowledge of problem instances' meta-features and algorithms' historical performances, many Machine Learning (ML) approaches are applied in AS to predict the algorithm performances on previously unseen problem instances [84]. Since the AS problem has a similar input structure and output target as a usual user-item recommendation task, the CF idea from RS is also applicable for AS scenarios. In this chapter, we borrow the learning to rank framework from RS and embed the bi-linear factorization to model the algorithms' performances in AS. Bi-Linear Learning to Rank (BLR) maps the problem meta-feature matrix to the algorithms performance matrix, and model the algorithms ranking in a list-wise probabilistic way. To counter the accuracy bias in AS evaluation, multi-objective metric A3R is applied to balance accuracy and inference time during the evaluation of algorithm selection approaches. BLR outperforms other benchmark approaches concerning A3R, especially when expanding the candidates range to $TOPK$. The necessity and benefit of the $TOPK$ expansion of the algorithms candidates is discussed.

5.1 Introduction

In the AS domain, for both computational complexity (e.g. SAT and CSP problems) and machine learning scenarios, the number of problem instances can be infinite, while a bunch of new algorithms are created for solving problem instances every year. Traversing with brute force on all the algorithms generates the exact performance of all problem instances and help select best algorithm precisely, though it's often time consuming. To speed up the selection process, AS approaches like Average Ranking (AveR), SB [158, 159] and ML


Algorithm Selection Problem
Can be seen as user-profiling features in recommender system.
Meta features of the problem instances
L meta features
M problem instances
Can be seen as user-item interaction matrix in recommender system.
Algorithm performances on problem instances
N algorithms
Learn the mapping
Performance Prediction as Cold start condition:
when new problem instance comes in the scenario.
New problem Instances with L meta features
Apply the mapping to predict the performances
Which one should be ranked as optimal algorithm with top1 performance?

0.4	10	7.7	0.1	67	1.5
0.3	15	5.5	0.2	78	1.7
0.7	22	2.5	0.1	96	3.2
0.8	33	5.4	0.6	88	5.4
0.1	14	3.3	0.2	78	3.3
0.2	25	6.8	0.8	90	2.6
0.5	11	9.8	0.2	91	1.6
0.9	44	3.4	0.3	78	6.1
0.2	47	3.4	0.5	54	6.8

0.3	0.5		1.0		0.2		1.2	3.2	0.4
10		2.3	10	9		22	5.5		2.2
	0.7	4.5	20	23	0.5	1.5		3.3	
1.5				17			2.8		4.2
10	0.9	2.8			15	0.8	3.2	5.4	
	0.8			18		0.9		8.9	11
1.2		5.9	15			7.8			
	2.0		23	19			1.5	10	20
4.5		2.7			7.5			5.6	4.5

0.9	45	5.4	0.7	77	9.0

?	?	?	?	?	?	?	?	?	?

Figure 5.1: Algorithm selection as recommendation system. Learning a model which maps the given meta features matrix (which size is $M \times L$) to the performance matrix (which size is $M \times N$). The inference of the new problem instance is like making a recommendation for a user in the cold-start phase.

meta-learning [77] have been proposed by researchers [160]. These AS approaches give a fast prediction on the potentially well-performing algorithms and make recommendations. AveR and SB only rely on the known performances (landmark features). ML approaches make use of both meta-features of problem instances and algorithms' past performances on the known problem instances to learn the prediction models. Many classic ML methods (like Support Vector Machine (SVM) and Random Forest (RF)) have been tried on the AS problem [77, 81, 91] and have proven to be powerful on some prediction tasks.

In Fig. 5.1, a typical AS problem is represented. Meta features of problem instances are fully given as the full matrix on left hand side, while performances of solvers(algorithms) applied on known problem instances form the performance matrix on the right hand side. The mapping function from meta features to the performances is expected to be learned. Given a new problem instance, the performance prediction fully relies on the meta-features vector. This full reliance makes the prediction task in AS similar to the *cold start* condition in RS. When looking at the blocks with stars in Fig 5.1, standing from the view of RS, the problem instance meta-feature input can be replaced by usual user profiling features like age, working field, preference category etc.. And the performance matrix can be associated with user rating or implicit feedback matrixRS. Therefore, the approaches used in RS can also be attempted in AS problem.

The terminologies used in AS, ML and RS occasionally overlap, we distinguish these terminologies in Table 5.1 to avoid misunderstanding. The definitions in the table are mostly based on the work of Bao et al. [161].

Table 5.1: Terminologies and explanations.

Term	Meaning
Scenario	An AS scenario is dedicated to solve a specific type of problem (e.g. Traveling Salesman Problem (TSP) in computational complexity domain). It is comprised of a set of problem instances, their meta-features and some performances of algorithms on the instances.
Problem Instance	One instance to be solved in a scenario, e.g. a complete graph in TSP.
Meta features	Descriptive features of a specific problem instance, like the number of edges, the number of nodes etc. in a complete graph for TSP problem.
Algorithm	Algorithm or heuristic (e.g. generic algorithm in TSP) which can successfully solve some of the problem instances in the designated scenario.
Solver	Alias for algorithm in some problems like SAT.
Approach	The method used to select the potential optimal algorithm candidates for problem instances in a specific scenario.
Predictor	The method that predicts the performance of the algorithms on a problem instance.
Selector	The method used to select the potential optimal algorithms based on their predicted performances.
Algorithm Candidate Set	A subset of algorithms from a scenario, provided by the selector as proposal for being the optimal ones for a specific problem instance.
Performance	A measurement representing of how well an algorithm solves a problem instance, e.g. *runtime*.

Evaluation Metric	The evaluation criteria to measure the selection effect of an approach in a scenario, e.g. Success Rate (SUCC), Miss-Classification Penalty (MCP).

Concerning applying RS approaches in AS problems, there are some open research problems:

RP5.1 How to model the mapping from meta-feature matrix to performance matrix directly? In AS, both problem meta-features and algorithm performance information are utilized for modeling. The benchmark approaches (will be introduced in Section 5.2) conduct models either by multi-model training for a specific algorithm or through one-hot encoding on algorithm performance. However, multi-models training cause extra time cost for inference, while one-hot encoding leads to extra sparsity in the training data. A model skipping these intermediate steps and creating the mapping directly improve the efficiency.

RP5.2 How to balance the AS effect when both prediction accuracy and inference time are taken into account in the evaluation process? To measure the effect of AS approaches, prediction error and precision are the standard evaluation criteria. However, problem instances and algorithms accumulate in specific AS scenario, AS approaches need to update the model accordingly. Thus inference time also reflects the effect of an AS approach from the perspective of model updating and predicting efficiency. A proper solution is needed to balance the accuracy based approach and time cost efficiency in the evaluation.

RP5.3 Whether we can expand the selection range thus benefit from the cumulative optimal algorithm? In most AS challenges [79, 80], only the optimal predicted algorithm is chosen for the evaluation, and this narrow set of candidate set reduces the chance of finding the optimal algorithm. If the $TOP1$ evaluation is expanded to $TOPK$, the cumulative performance gain guarantees some AS approaches reaching higher accuracy. Whether such expansion brings advantages to AS needs to be discussed and studied.

In order to address the research problems raised above, we construct the following studies in this chapter:

1) We propose BLR to involve both problem instance meta-features and performance matrix into one L2R framework. The mapping matrix W in the model accomplishes the mapping from meta-feature to the performance matrix in a straightforward way. The probabilistic assumption on the ranking solve the randomness modeling of the performance value in the algorithm-problem instance interaction matrix. We illustrate the good performance of BLR by comparing it with other benchmark AS approaches in the designed experiments.

2) A3R was proposed as a ranking measure for the algorithms in ML meta-learning. It incorporates both accuracy oriented metric and time cost metric into a single, scalar

evaluation metric. We apply A3R as the evaluation metric for the general AS task, in order to balance the accuracy and inference time for measuring AS approaches. Being measured with A3R, BLR outperforms other approaches in terms of this trade-off.

3) In order to demonstrate the performance variance given different $TOPK$ candidates set range setting, we compare the approaches ranking distribution for the specific K selection. To observe the benefit brought from the K expansion, the AS effect increasement of the cumulative optimal performance is also listed and compared.

The chapter is structured as follows: basic methodologies, benchmark approaches and concrete modeling steps of BLR are introduced in Section 5.2. In Section 5.3, we review the evaluation metrics frequently used in AS tasks and propose A3R as the trade-off metric for accuracy and inference time. Section 5.4 compares the performance of the BLR with other benchmark approaches, with the emphasis on A3R. Subsequently, Section 5.5 explains the necessity of applying $TOPK$ expansion of the candidate selection while evaluating AS approaches. Finally, Section 5.6 draws the conclusion and gives an outlook to the future work.

5.2 Algorithm Selection - Methodologies

In AS scenarios, regarding one problem instance, the targets of the prediction are the performances of multiple algorithms, rather than a single label or a numerical value. The performance scores of algorithms are estimated with this multi-target prediction task. The algorithms are thereby selected according to the predicted scores. Among many solutions, there are three conventional ways to design AS approaches: 1) purely relying on statistics of algorithms' historical performances (e.g. select the algorithm with the "best performance" on all the known problem instances); 2) algorithm specific performance prediction: building the predicting model for each algorithm given the meta-features of the problem instances individually, fit the model and do the inference specifically; 3) algorithm indicators' one-hot conversion: horizontally concatenate the problem instance meta-feature matrix and algorithm appearance one-hot matrix to form the input matrix as the input for the general prediction function. In this section, we first introduce the benchmark approaches which follow these three ways of approach design. Subsequently, we propose BLR, which doesn't need multi-model training and one-hot conversion to complete the AS model creation.

5.2.1 Benchmark Approaches

Targeting diverse AS scenarios[1], we separate the benchmark approaches in the community into three groups according to the data transformation ways mentioned above.

[1]http://coseal.github.io/aslib-r/scenario-pages/

- **Statistics of historical performances**: AS approaches based on statistics of historical performances don't rely on any ML model assumption of meta features, whereas use the performance matrix to gain the statistics directly. **Virtual Best Selector** and **Single Best** are two traditional benchmark approaches in AS obeying this rule. **Virtual Best Selector** is the ground truth of the algorithms performances. The rankings of algorithms in Virtual Best Solver (VBS) are the true ranks used to compare with the predicted list. The algorithm picked by VBS is with the highest ground-truth performance for every problem instance. The evaluation result of the VBS list is the upper bound of all other AS approach. **Single Best** is the most classic algorithm selection baseline approach. It selects the algorithm whose mean performance is the best through all the problem instances in the training set.

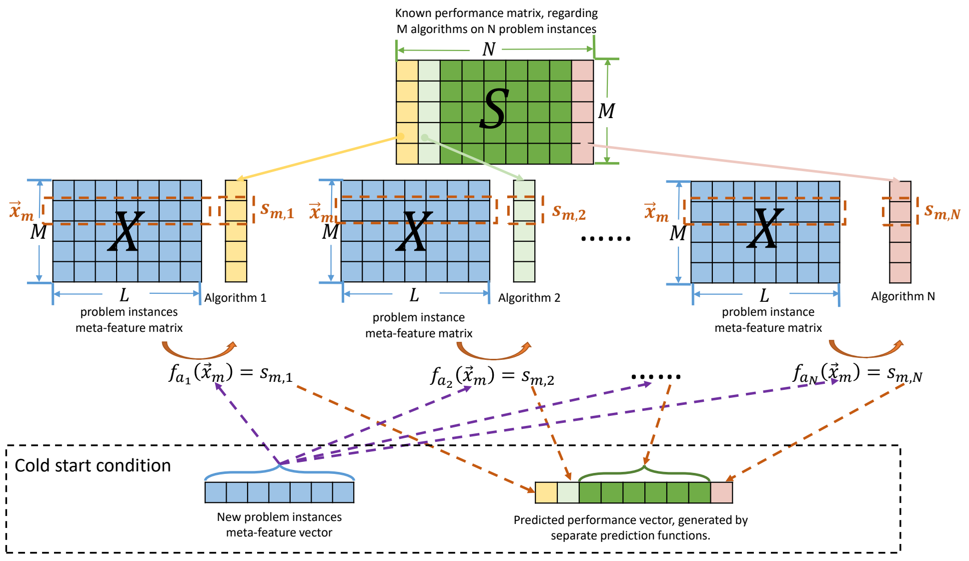

Known performance matrix, regarding M algorithms on N problem instances
N
S
M
$\vec{x}_m$
M
X
$s_{m,1}$
L
Algorithm 1
problem instances meta-feature matrix
$\vec{x}_m$
M
X
$s_{m,2}$
L
Algorithm 2
problem instance meta-feature matrix
......
$\vec{x}_m$
M
X
$s_{m,N}$
L
Algorithm N
problem instance meta-feature matrix
$f_{a_1}(\vec{x}_m) = s_{m,1}$
$f_{a_2}(\vec{x}_m) = s_{m,2}$
......
$f_{a_N}(\vec{x}_m) = s_{m,N}$
Cold start condition
New problem instances meta-feature vector
Predicted performance vector, generated by separate prediction functions.

Figure 5.2: Algorithm based separated modeling. For each algorithm a_n, regarding its performances $s_{:,n}$ (n_{th} column in performance matrix S), we learn the mapping function $f_{a_n}(\vec{x}_m) = s_{m,n}$ which infers the meta-feature vectors in X to $s_{:,n}$. N mapping functions are learned for N algorithms. Under cold start condition (in the bottom dashed box), for a new problem instance m, N mapping functions $f_{a_n}(\vec{x}_m)$ are applied on N algorithms. The recommended algorithm list is ordered according to the predicted scores.

- **Algorithm specific performance prediction**: The algorithm specific performance prediction process is explained in Fig. 5.2. For each algorithm, a single prediction model is trained based on problem instances' meta-features and the performances of this specific algorithm. When a new problem instance shows up, N prediction models are used to estimate the performances for the N algorithms separately. The following AS approaches adopt the algorithm specific performance prediction process. **Separated Linear Regressor** trains linear regressor for candidate algorithms separately. When a new problem instance comes, performance prediction on algorithms depends on every fitted linear model. **Separated Random Forest Regressor** fits RFs for every algorithm. During the inference phase, N RFs are called separately to generate predictions for N algorithm individually. Similarly, **Separated Gradient Boosted Regression Trees (XGBoost)** uses gradient boosted trees to learn the performance predictor, every individual algorithm owns a XGBoost model and infer the new performance value based on its own XGBoost model. In spite of the model specialty of this group of AS approaches, long inference time and heavy memory consumption are the main disadvantage of these approaches.

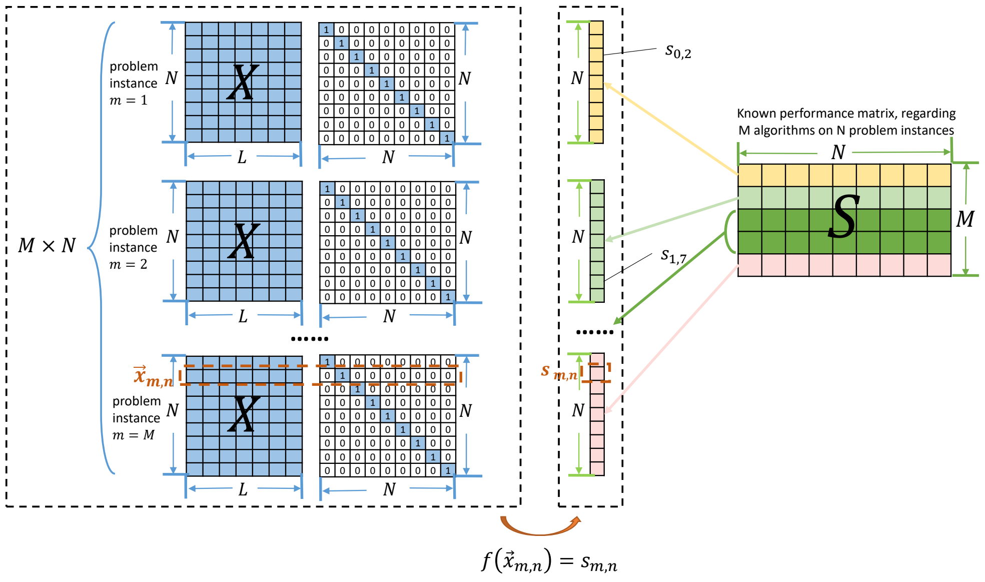


Figure 5.3: Algorithm one-hot conversion. Converting the algorithm indicator to a one-hot vector, combine it with the problem instance meta-feature vector to form the training input $\vec{x}_{m,n}$ (line within the dashed block in brown) to predict the performance of a problem instance - algorithm pair $s_{m,n}$. The mapping function $f(\vec{x}_{m,n}) = s_{m,n}$ is learned from the stacked input features with dimension $((M \times N), (L + N))$ (as the dashed box on the left hand side). The rows in the performance matrix are transposed and stacked to form the performance column as the predicting target (shown inside the dashed box in the middle). When a new problem instance $m+1$ comes in, its meta-feature vector will be concatenated with all one-hot algorithms indicator vectors from algorithm indicators to form the input matrix, and mapping function f will map this matrix to the performance vector $\vec{s}_{m+1,:}$.

- **Algorithm One-hot Conversion**: Another group of AS approaches apply the one-hot conversion of the algorithms appearance indicator to form the AS input features. Regarding a new problem instance, concatenated vector of problem instance meta-feature and the algorithm indicator vector forms the input for the prediction model. Fig. 5.3 represents the conversion process. Though the single model brings in the simplicity, algorithms indicators one-hot conversion create extra sparsity for the input vector $\vec{x}_{m,n}$. The AS approaches following this conversion rules include: **One-hot Linear Regressor** training one linear predicting model with the flattened representation from the combination of problem instance meta-features and algorithms appearance indicators. Only one linear model is applied during inferencing for new problem instances. **One-hot RF Regressor** has each entry in the performance matrix as the regression target, with the $L + N$ dimensional features, only one RF is needed to fit the model. The model can infer any algorithm's performance with it's one-hot encoded appearance indicator. **One-hot XGBoost** also fits a single XGBoost model with $M \times N$ training samples, it is applicable for the performances inference for all the algorithms.

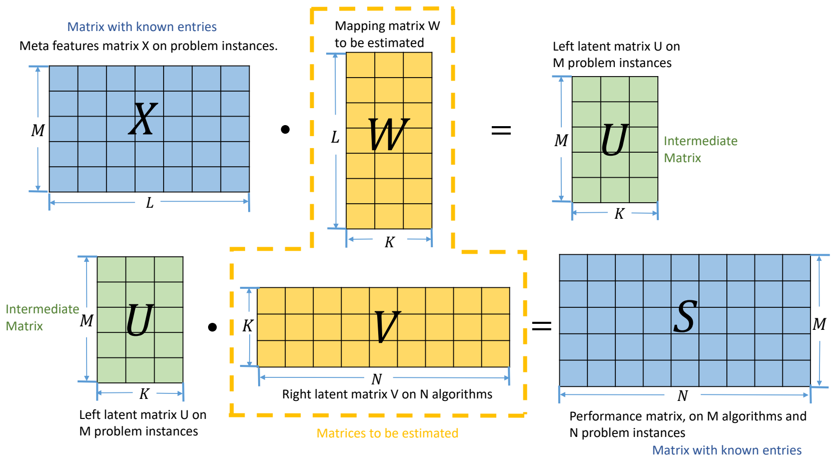

Matrix with known entries
Meta features matrix X on problem instances.
Mapping matrix W to be estimated
Left latent matrix U on M problem instances
X
M
L
W
L
K
U
M
K
Intermediate Matrix
Intermediate Matrix
U
M
K
V
K
N
S
M
N
Right latent matrix V on N algorithms
Left latent matrix U on M problem instances
Matrices to be estimated
Performance matrix, on M algorithms and N problem instances
Matrix with known entries

Figure 5.4: Bi-linear factorization graph given the known matrices. Problem instance meta-feature matrix X and performance matrix S are the targets in the factorization process. W is supposed as the weighted mapping matrix for input X. It projects X onto the intermediate left latent matrix U with K latent dimensions for M problem instances. The dot product of intermediate left latent matrix U and right latent matrix V yields the performance matrix S (in blue, known entries in the training set). Aside from the known and intermediate matrices, the unknown matrices W and V are to be estimated during the training process.

5.2.2 Bi-linear L2R

In AS scenarios, there are two matrices with known entries. One is the problem instance meta-feature matrix X, the other is the algorithm problem instance performance matrix S. The benchmark approaches mentioned in the above subsection solve the mapping from X to S via either multi-models training (time consuming) or algorithms' indicators' one-hot conversion (can sparsify the dataset). In order to avoid multi-models training and features one-hot conversion, we propose BLR to create the AS strategies. Assuming a bi-linear relationship between the problem instances' features and the target performance values, the decomposed multiplication process of the mapping from X to S is represented in Fig. 5.4. The inference process of the performance of new problem instances is depicted in Fig. 5.5.

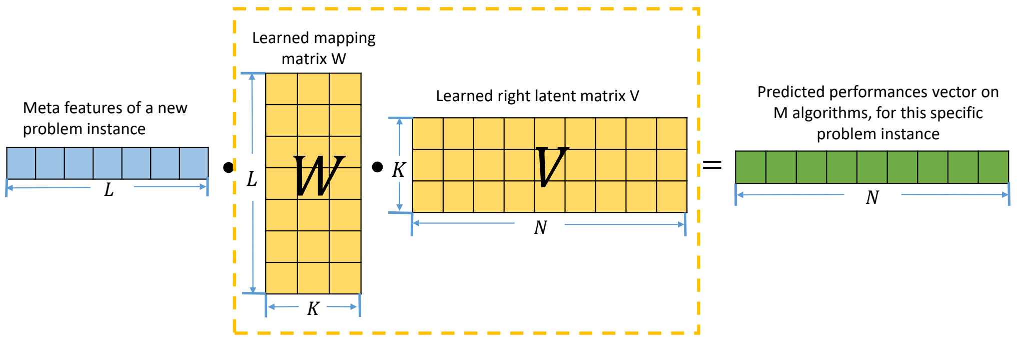

Meta features of a new problem instance
L
Learned mapping matrix W
L
W
K
Learned right latent matrix V
K
V
N
Predicted performances vector on M algorithms, for this specific problem instance
N

Figure 5.5: Algorithm selection as a cold start problem under Bi-linear Decomposition. W and V are the decomposed matrices after Bi-linear factorization from problem instance meta-feature matrix and performance matrix. When a new problem instance is introduced into a scenario with only its own meta-feature vector (on the left in blue), yet without any algorithm performance record. The continuous dot product on this meta-feature vector and the learned matrices W, V yields the full performance (on the right in green) vector regarding this new problem instance.

In algorithms' performances, uncertainty always exists. For computational complexity problems, like SAT and TSP, the algorithm's *runtime* performance can be different when the specific running environment alters. Based on the Bi-linear factorization assumption on the performance on the algorithm performances, we wrap the ranking of algorithms w.r.t. a specific problem instance in a probabilistic fashion. We assume the probability an algorithm ranked $TOP1$ for a problem instance is proportional to it's performance (or predicted performance) among all the algorithms. The cross entropy between the ground truth $TOP1$ probability vector $P_{\vec{r}_m}(r_{m,n})$ and the predicted $TOP1$ probability vector $P_{\vec{\hat{r}}_m}(\hat{r}_{m,n})$ (where r is the converted value of a performance value s) defines the loss and influence the optimization strategy.

Embedding bi-linear factorization in the L2R framework is the full idea of BLR. We refine the notations for BLR in Table 5.2. The modeling and learning of BLR is devided into four steps: 1) Performance scoring model function and corresponding rating converting function; 2) Loss function considering the ranking loss; 3) Gradient function for corresponding weights; and 4) Updating rule of the weights according to the specific optimization approach. The first two steps are introduced as follows in this section, while the gradient function and updating rules are explained in Subsection 5.2.3.

Table 5.2: Notations of Bi-linear Learning to Rank.

M	Number of problem instances in the training set.
N	Number of algorithms in the training set.
L	Number of meta features calculated for each problem instance.
K	Dimension number of the latent factor.
$\mathbf{S}_{M\times N}$	The performance matrix for N algorithms on M problem instances.
$s_{m,n}$	The performance value of algorithm n on problem instance m.
$\hat{\mathbf{S}}_{M\times N}$	The predicted performance matrix for N algorithms on M problem instances.
$\hat{s}_{m,n}$	The predicted performance value of algorithm n on problem instance m.
$\mathbf{X}_{M\times L}$	Values of L meta-features on M problem instances.
$x_{m,l}$	The l_{th} meta feature of problem instance m.
$\mathbf{W}_{L\times K}$	Bi-linear weight matrix which maps from L problem meta features to k dimensional latent feature space.
$w_{l,k}$	The mapping factor for the l_{th} meta feature on the k_{th} latent factor.
$\mathbf{U}_{M\times K}$	Matrix of K dimensional latent vector for M problem instances.
$u_{m,k}$	The k_{th} latent factor of problem instance m.
$\mathbf{V}_{N\times K}$	Matrix of K dimensional latent vector for N algorithms.
$v_{n,k}$	The k_{th} latent factor of algorithm n.
$\mathbf{R}_{M\times N}$	Matrix of performance values of N algorithms on M problem instances (A converted representation of $\mathbf{S}_{M\times N}$, which assigns better performing algorithms a higher value).

$r_{m,n}$	The converted rating value of algorithm n on problem instance m.
$\hat{\mathbf{R}}_{M\times N}$	Matrix of estimated performance ratings of N algorithms on M problem instances.
$\hat{r}_{m,n}$	The estimated converted rating value of algorithm n on problem instance m.
$P_{\vec{r_m}}(r_{m,n})$	Given the known performance rating vector $\vec{r_m}$, the probability that algorithm n is ranked at top 1 regarding the m_{th} problem instance.
$P_{\vec{\hat{r}}_m}(\hat{r}_{m,n})$	Given the *estimated* rating vector $\vec{\hat{r}}_m$, the probability that algorithm n is ranked at top 1 regarding the m_{th} problem instance.

5.2.2.1 Model Function

In BLR, given the problem instance m and algorithm n, we predict the performance score as $\hat{s}_{m,n}$ in the Eq. 5.1. $\hat{s}_{m,n}$ is supposed to be the result of the product from the latent vector of m_{th} problem instance $\vec{u}_m$ and the latent vector of n_{th} algorithm $\vec{v}_n$. $\vec{u}_m$ is comprised from original meta feature vector $\vec{x}_m$ and $\mathbf{W}$. Thus entries $\vec{v}_{n,k}$ and $\vec{w}_{l,k}$ in the matrices are the ones to be estimated.

In AS, the preferred sorting order on performance values depends on the choice of the target performance. E.g. if the performance metric is *runtime*, a lower value is better. Yet, if *accuracy* is the targeted performance metric, a higher value is preferred. For the simplicity of calculating the list-wise ranking loss, we define a conversion function $r = f(s)$ to make descending order preferable for all the rating values r. The converted rating value r is the optimization unit in the ranking loss function. In this paper, we simply define $f(s)$ as Eq. 5.2.

$$\begin{aligned}\hat{s}_{m,n} &= \vec{u}_m \cdot \vec{v}_n^T \\ &= \vec{x}_m \times \mathbf{W} \cdot \vec{v}_n^T \\ &= \sum_{k=1}^{K}\left(v_{n,k} \cdot \sum_{l=1}^{L} x_{m,l} \cdot w_{l,k}\right)\end{aligned} \tag{5.1}$$

$$f(s) = \begin{cases} s & \text{higher performance value is preferred} \\ -s & \text{lower performance value is preferred} \end{cases} \tag{5.2}$$

5.2.2.2 List-wise Loss Function

The performance scores of algorithms on specific problem instance are with measuring noises. Thus with the performance value scoring function, the probability that an algorithm being ranked top-one can be modeled. The normalized top-one probability representation has been proposed in the L2R domain to represent the list-wise ranking loss [40]. Regarding a single problem instance, the top-one probability for the same algorithm is different between the ground truth performances list and the predicted performances list. As defined in Eq. 5.3, for a problem instance m, with the rating vector $\vec{r}_m$ (converted version of the performance vector), the top-one probability for each algorithm n is normalized in the form of $P_{\vec{r}_m}$.

$$P_{\vec{r}_m}(r_{m,n}) = \frac{\varphi(r_{m,n})}{\sum_{n=1}^{N} \varphi(r_{m,n})} \tag{5.3}$$

The exponential function is applied as the concrete form for monotonically increasing function φ in Eq. 5.3. $P_{\vec{r}_m}$ can be represented as Eq. 5.4, which is in the same shape of Softmax function representation. The exponential function makes probability distribution more gathered around the position of the largest input values, thus make the $TOP1$ item easier to distinguish.

$$P_{\vec{r}_m} = \frac{\exp(r_{m,n})}{\sum_{n=1}^{N} \exp(r_{m,n})} \tag{5.4}$$

To represent the list-wise ranking loss per problem instance, the cross entropy is calculated between the top-one probability from the predicted rating list $\vec{\hat{r}}_m$ and the ground truth rating value list $\vec{r}_m$. For each problem instance m, the point-wise loss for algorithm n is formulated as Eq. 5.5. Considering the probabilities normalization is calculated under the same scale for a problem instance m, the per instance list-wise loss L_m is defined as the summation of the point-wise loss inside this list, as shown in Eq. 5.6. Here L_m is the list-wise ranking loss between the ground truth list and the predicted list. The total loss on the whole m problem instances is defined in Eq. 5.7, in which L2 regularization is applied to avoid over-fitting.

$$L_{m,n} = -P_{\vec{r}_m}(r_{m,n}) \ln P_{\vec{\hat{r}}_m}(\hat{r}_{m,n}) \tag{5.5}$$

$$L_m = -\sum_{n=1}^{N} P_{\vec{r}_m}(r_{m,n}) \ln P_{\vec{\hat{r}}_m}(\hat{r}_{m,n}) \tag{5.6}$$

$$L = \sum_{m=1}^{M} L_m + \frac{\lambda}{2}\left(\|\mathbf{W}\|_F^2 + \|\mathbf{V}\|_F^2\right) \tag{5.7}$$

5.2.3 Gradient and Updating Rules in Bi-linear L2R

In the BLR model, in order to approximate the weighting matrix $\mathbf{W}$ and latent matrix $\mathbf{V}$ to minimize the loss function defined in the Subsection 5.2.2, we calculate the gradient of the loss function and use the updating rule as described in the this subsection.

- **Gradient Calculation** With the loss function L, when using gradient descent as the optimizer, the gradient calculation concerning meta features mapping weight matrix $\frac{\partial \hat{s}_{m,n}}{\partial \mathbf{W}}$ and algorithm latent vectors matrix $\frac{\partial \hat{s}_{m,n}}{\partial \vec{v}_n}$ should be provided accordingly. Since the loss function is factorized layer by layer through out model function, converter function, top-one probability function and cross entropy function, we use chain rule to calculate the gradient correspondingly. For L, its partial differential over $w_{l,k}$ and $v_{n,k}$ can be factorized in the similar way as Eq. 5.8 and Eq. 5.9 separately.

$$\frac{\partial L}{\partial w_{l,k}} = \sum_{m=1}^{M}\sum_{n=1}^{N} \frac{\partial L_{m,n}}{\partial P_{\vec{\hat{r}}_m}(\hat{r}_{m,n})} \frac{\partial P_{\vec{\hat{r}}_m}(\hat{r}_{m,n})}{\partial \hat{r}_{m,n}} \frac{\partial \hat{r}_{m,n}}{\partial \hat{s}_{m,n}} \frac{\hat{s}_{m,n}}{\partial w_{l,k}} + \lambda w_{l,k} \tag{5.8}$$

$$\frac{\partial L}{\partial v_{n,k}} = \sum_{m=1}^{M}\sum_{n=1}^{N} \frac{\partial L_{m,n}}{\partial P_{\vec{\hat{r}}_m}(\hat{r}_{m,n})} \frac{\partial P_{\vec{\hat{r}}_m}(\hat{r}_{m,n})}{\partial \hat{r}_{m,n}} \frac{\partial \hat{r}_{m,n}}{\partial \hat{s}_{m,n}} \frac{\hat{s}_{m,n}}{\partial v_{n,k}} + \lambda v_{n,k} \tag{5.9}$$

For each $L_{m,n}$, the intermediate calculation steps for deviation according to chain rule can be derived as following for each $L_{m,n}$:

$$\frac{\partial L_{m,n}}{\partial P_{\vec{\hat{r}}_m}(\hat{r}_{m,n})} = -P_{\vec{r}_m}(r_{m,n}) \frac{1}{P_{\vec{\hat{r}}_m}(\hat{r}_{m,n})}$$

$$\frac{\partial P_{\vec{\hat{r}}_m}(\hat{r}_{m,n})}{\hat{r}_{m,n}} = \frac{\partial}{\partial \hat{r}_{m,n}} \frac{\exp \hat{r}_{m,n}}{\sum_{n=1}^{N} \exp \hat{r}_{m,n}}$$

$$= \frac{\exp(\hat{r}_{m,n})}{\sum_{n=1}^{N} \exp(\hat{r}_{m,n})} - \frac{\exp(\hat{r}_{m,n})^2}{(\sum_{n=1}^{N} \exp(\hat{r}_{m,n}))^2}$$

$$\frac{\partial \hat{r}_{m,n}}{\partial \hat{s}_{m,n}} = -1 \tag{5.10}$$

$$\frac{\partial \hat{s}_{m,n}}{\partial w_{l,k}} = x_{m,l} v_{n,k} \tag{5.11}$$

$$\frac{\partial \hat{s}_{m,n}}{\partial v_{n,k}} = \sum_{l=1}^{L} x_{m,l} w_{l,k} \tag{5.12}$$

For the last step, which returns the partial differential $\hat{s}_{m,n}$ over $w_{l,k}$ and $v_{n,k}$, we can broadcast it in the vectorized way like Eq. 5.13 and Eq. 5.14:

$$\frac{\partial \hat{s}_{m,n}}{\partial \mathbf{W}} = \vec{x}_m \otimes \vec{v}_n \tag{5.13}$$

$$\frac{\partial \hat{s}_{m,n}}{\partial \vec{v}_n} = \vec{x}_m \times \mathbf{W} \tag{5.14}$$

- **Updating Rule**

  Having known the partial differential of the chain rule, we can update the weight matrix $\mathbf{W}$ and algorithm latent matrix $\mathbf{V}$ by the following updating rule Eq. 5.15 and 5.16, where η is the learning rate.

$$\mathbf{W}^t = \mathbf{W}^{t-1} - \eta \frac{\partial L}{\partial \mathbf{W}^{t-1}} \tag{5.15}$$

$$\vec{v}_n = \vec{v}_n^{t-1} - \eta \frac{\partial L}{\partial \vec{v}_n^{t-1}} \tag{5.16}$$

  Given the list-wise defined entropy loss and gradient, the parameters updating rule for W and V should also be list-wise. The updating batch is based on performances vector of an algorithm list targeting a problem instance m, rather than each performance point. The weights are updated in a stochastic way, and the stochastic updating rule is defined in Eq. 5.17 and Eq. 5.18:

$$\begin{aligned}\mathbf{W}^t &= \mathbf{W}^{t-1} - \eta \frac{\partial L_m}{\partial \mathbf{W}^{t-1}} \\ &= \mathbf{W}^{t-1} - \sum_{n=1}^{N} \left(\eta \frac{\partial L_{m,n}}{\partial \mathbf{W}^{t-1}} \right)\end{aligned} \tag{5.17}$$

$$\vec{v}_n = \vec{v}_n^{t-1} - \eta \frac{\partial L_m}{\partial \vec{v}_n^{t-1}} \tag{5.18}$$

5.2.4 Discussion

In this section, we discuss the methodologies for AS problems. Solving an AS problem in our specific setting is equivalent to fitting the mapping from meta-features matrix to the performance features matrix. Conventional approaches either need to train separate models for specific algorithms or add extra one-hot encoding step for generating the features. In order to skip these intermediate steps, we propose BLR to do the direct mapping, and solve the **RP5.1 How to model the mapping from meta-feature matrix to performance matrix directly?** mentioned in the introduction.

5.3 Evaluation Bias and Trade Off

In the AS community, SUCC, Penalized Average Runtime Score (PAR10) and MCP are the standard evaluation metrics. However, these metrics mainly focus on whether the approach can precisely select the algorithms with better performance. Other factors like selection/inference time of AS approaches are not frequently taken into account. To balance the trade-off between the accuracy oriented bias and inference time cost, multi-objectives evaluation metric deserves the research interest.

In this section, we first introduce the conventional evaluation metrics in AS, then propose a multi-objective evaluation in the AS scenarios to address the evaluation bias. In the end, the relevant datasets in the AS scenarios for the follow up evaluations are also summarized.

5.3.1 Accuracy Oriented Evaluation Metrics

In this chapter, four classic accuracy oriented benchmark evaluation metrics for comparing the approaches in AS relevant problems have been introduced. These metrics prone to select AS approach which correctly tell the algorithms with the best performance entries. Obeying the conventional candidate selection criteria, the selection range of algorithms is limited to $TOP1$ from the predicted list. Aside from that, we also give the explanation assuming $TOPK$ selection for each evaluation metric.

Success Rate (SUCC) stands for the average solved ratio of the selected algorithm per problem instance across the test set. For the $TOP1$ selection criteria, the solved ratio is only calculated for the algorithm with the best predicted performance. For the $TOPK$ selection criteria, the average is calculated on the success rate through all the top K selected algorithms.

Penalized Average Runtime Score (PAR10) is the penalized version for the actual *runtime* of the selected algorithm. If the selected algorithm took longer than the time hold threshold, it is instead counted as ten times the amount of time of the threshold. Otherwise the actual *runtime* is directly used. With $TOP1$ selection criteria, the penalty is only applied on the best ranked algorithm in the predicted list. For PAR10@K, the penalty will be applied on the algorithm with the shortest actual *runtime* in the $TOPK$ predicted list.

Miss-Classification Penalty (MCP) compares the runtime difference between the actual *runtime* of the predicted best algorithm and the VBS with the $TOP1$ selection criteria. For the $TOPK$ case, the algorithm with the lowest actual *runtime* in the $TOPK$ predicted list is chosen to be compared with the *runtime* of VBS. The algorithm selected by VBS has an MCP value of zero.

Mean Average Precision (MAP) measures the mean average precision of the $TOPK$ predicted algorithms vs. the $TOPK$ ranked algorithms with the ground truth performance. MAP for $TOPK$ algorithms in the predicted list is calculated in the same way as MAP@K.

5.3.2 Multi-objective Evaluation Metrics

Accuracy oriented evaluation metrics SUCC and MAP comply with the rule *the higher the better*, while for the runtime oriented metrics like MCP and MAP, *the lower the better*. Multi-objective evaluation metric Adjusted Ratio of Root Ratios (A3R) combines accuracy and runtime into a single score. Abdulrahman, Salisu et al. first introduced A3R in AutoML [162, 163] as the ranking basis for algorithms w.r.t. a dataset in AutoML. A3R balances the precision and the runtime of the selected algorithm. As Eq. 5.19 shows, when applying algorithm a_p on dataset d_i, $SR^{d_i}_{a_p}$ stands for the success rate and $T^{d_i}_{a_p}$ represents the runtime. A reference algorithm a_q is chosen, to standardize the relative success rate across all the algorithms as ratio $SR^{d_i}_{a_p}/SR^{d_i}_{a_q}$. The equivalent ratio for runtime is represented as $T^{d_i}_{a_p}/T^{d_i}_{a_q}$. The combined metric takes success rate ratio as advantage, while the time ratio as disadvantage. Since the runtime ratio ranges across more magnitudes than the success rate does, N_{th} root on the denominator of Eg. 5.19 enables the re-scaling of the running time ratio, and turns the $A3R$ to a reasonable value range. A3R is used to measure the comprehensive quality running an algorithm on the dataset.

$$A3R^{d_i}_{a_p a_q} = \frac{SR^{d_i}_{a_p}/SR^{d_i}_{a_q}}{\sqrt[N]{T^{d_i}_{a_p}/T^{d_i}_{a_q}}} \tag{5.19}$$

$$A3R(ACC)^{s_i}_{a_p a_q} = \frac{ACC^{s_i}_{a_p}/ACC^{s_i}_{a_q}}{\sqrt[N]{T^{s_i}_{a_p}/T^{s_i}_{a_q}}} \tag{5.20}$$

$$A3R(TC)^{s_i}_{a_p a_q} = \frac{\sqrt[M]{TC^{s_i}_{a_q}/TC^{s_i}_{a_p}}}{\sqrt[N]{T^{s_i}_{a_p}/T^{s_i}_{a_q}}} \tag{5.21}$$

In this chapter, we borrow the idea of A3R from AutoML, and apply it as the evaluation metrics for the approaches in the AS scenario. We replace d_i with s_i (the i_{th} scenario), keep a but note as approach in Eq. 5.20. For accuracy based metrics like SUCC and MAP, we apply their values ACC to substitute SR in the Eq. 5.19. For runtime based metrics TC, lower values gain more preference. Thus the inverse ratio $TC^{s_i}_{a_q}/TC^{s_i}_{a_p}$ is instead used in the numerator as Eq. 5.21. Considering the runtime can also span several magnitudes, M_{th} root is applied on the numerator for re-scaling. In the following experiments, we use Eq. 5.20 and Eq. 5.21 to evaluate the AS effect with multi-objectives.

5.3.3 Experiments Set Up

Targeting on the evaluation protocols, we design the experiments to study: 1) The algorithm selection effect of the proposed BLR approach compared with other benchmark approaches; 2) the AS effect of the proposed approaches when taking both *accuracy* and *inference time* into consideration; 3) the benefits of expanding the candidates set selection range. The datasets to conduct the research in this chapter are introduced as following.

For the computational complexity AS scenarios, the Algorithm Selection Library (ASLib) released by COnfiguration and SElection of ALgorithms (COSEAL) [2] research group provides the most complete and standardized datasets. In the experiments, we fetch the following scenarios from ASLib: *ASP-POTASSCO*, *BNSL-2016*, *CPMP-2015*, *CSP-2010*, *CSP-MZN-2013*, *CSP-Minizinc-Obj-2016*, *GRAPHS-2015*, *MAXSAT12-PMS*, *MAXSAT15-PMS-INDU*, *PROTEUS-2014*, *QBF-2011*, *QBF-2014*, *SAT11-HAND*, *SAT11-INDU*, *SAT11-RAND*, *SAT12-ALL*, *SAT12-INDU*, *SAT12-RAND*, *SAT15-INDU*, *TSP-LION2015*. In all of these computational complexity AS scenarios, *runtime* is the main performance metric. In each scenario, the dataset comprises of algorithms' performances on problem instances, problem instances meta-features run status, and feature values. The standardized datasets make the experiments evaluation results among many scenarios comparable.

In each AS scenario from the ASLib, we split the dataset into 10 folds, apply cross validation on the 9-folds to find the best hyper-parameter setting for each approach. With the best selected hyper parameters, all approaches are trained again on the whole 9-folds dataset thus the fitted models are acquired. These models are used to do the inference on the last fold (test set) to be evaluated.

5.3.4 Discussion

In AS problems, accuracy oriented evaluation metrics are the main measuring method. Its dominant role makes other performance effect being ignored. In this section, we propose apply the multi-objective evaluation metric A3R to balance the trade-off between the approach prediction accuracy and inference time. Therefore, the research problem **RP5.2 How to balance the AS effect when both prediction accuracy and inference time are taken into account in the evaluation process?** is solved within this section.

5.4 BLR: Balancing Accuracy and Inference Time

We compare the AS effect of BLR with other benchmark approaches under the five evaluation metrics introduced in the last section. For BLR model, latent dimension K, learning rate η, regularizer λ are the hyper parameters to be tuned during cross validation.

[2]https://www.coseal.net/

Since the optimization target of BLR decomposition is not convex, the trained model is sensitive to the initialization of the entries in the latent matrices. Thus the best initialization state is also determined in the cross validation phase. To speed up the convergence of the BLR, we use Stochastic Gradient Descent (SGD) instead of Gradient Descent (GD) as optimization method. Given the vibrated loss value on SGD, we tell the convergence of BLR model with at least 5 successive increases on the loss detected during the optimization.

In this section, we set K as 1 and 3 to compare the difference in the evaluation result more specially for the performance of BLR. In order to illustrate the influence of TOPK candidates selection criteria on the cumulative evaluation result, we select the representative scenarios and draw the performance curve for BLR and other benchmark approaches at different ranking positions. Under the condition K set to 3, we investigate the accuracy and inference time trade off for all the approaches mentioned, and found that BLR represents the competitive role concerning this trade off measurement.

Table 5.3: Scenarios and evaluation metric under which Bi-linear L2R is measured as top 3 among all the benchmark approaches, when the candidate set size is set to ONE.

Scenario Name	Evaluation Metric	Rank
CSP-Mininzic-Obj-2016	SUCC	1
GRAPHS-2015	SUCC	3
PROTEUS-2014	MCP	1
PROTEUS-2014	PAR10	1
SAT11-INDU	SUCC	2
SAT12-RAND	MAP	2
SAT15-INDU	SUCC	1
SAT15-INDU	MCP	2
SAT15-INDU	PAR10	3
TSP-LION2015	MAP	1

5.4.1 BLR performance with different candidates selection criteria

Performance with TOP 1 Selection Criteria First we apply the conventional $TOP1$ candidates selection in the evaluation, and observe under what circumstances BLR performs better. In Table 5.3, AS scenario and evaluation metric combination are listed per row. These are the cases BLR is ranked among the best 3 compared with other benchmark approaches. More specifically, in *CSP-Mininzic-Obj-2016* and *SAT15-INDU* regarding *success rate*, in *PROTEUS-2014* concerning MCP and PAR10, in *TSP-LION2015* in terms of MAP, BLR is ranked as top1. These competitive performances verify that BLR can also be considered as a benchmark approach in some AS scenarios.

Performance with expanded TOP 3 Selection Criteria The cumulative best performance varies a lot even considering single AS approach, thus the rank of approaches also changes when considering different expansion degrees. For BLR, aside from the conventional $TOP1$ candidates selection criteria, we observe its rankings under $TOP3$ selection. In Table 5.4, we list the conditions (combinations of scenario and evaluation metric) where BLR is evaluated as competitive (ranked in top 3). BLR can still perform well in some specific scenarios. When being compared with Table 5.3, the advancing performances of BLR doesn't hold consistent between $TOP1$ and $TOP3$ candidates selection in most scenarios. Only in scenarios *GRAPHS-2015* and *TSP-LION2015*, BLR shows competitive role in both $TOP1$ and $TOP3$.

Table 5.4: Scenarios and evaluation metric under which Bi-linear L2R is measured as top 3 among all the benchmark approaches, when the candidate set size is set to THREE.

Scenario Name	Evaluation Metric	Rank
CPMP-2015	MAP	3
CSP-2010	SUCC	2
CSP-2010	MAP	2
GRAPHS-2015	MCP	3
MAXSAT12-PMS	MCP	3
MAXSAT12-PMS	PAR10	2
PROTEUS-2014	MAP	3
SAT11-HAND	MCP	3
SAT11-HAND	PAR10	3
TSP-LION2015	MAP	1

5.4.2 Cumulative Performance in TOPK expansion

If parallel processing on the candidate algorithms is considered, we can broaden the range of candidates selection to increase the chance of finding the best algorithm without extra time consumption. Thus if the cumulative best performances of approaches increase drastically at first several predicted positions, it's proper to consider $TOPK$ expansion for the predicted list. We first observe the cumulative best performance along the $TOPK$ position elapse in some scenarios. For *SAT11-HAND*, *PROTEUS-2014* and *MAXSAT12-*

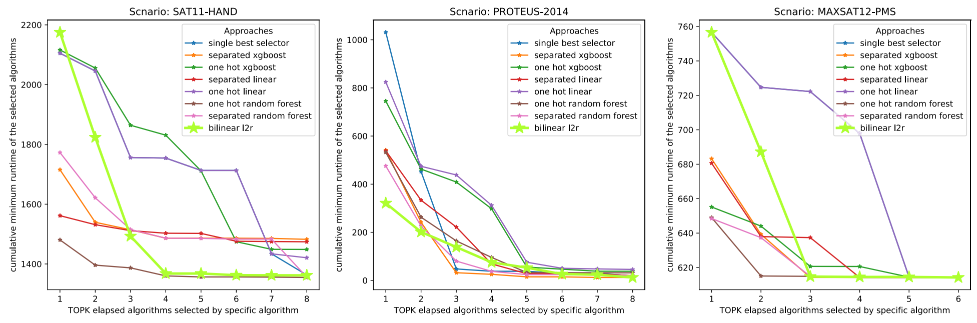


Figure 5.6: Cumulative minimum runtime (average across all the predicted problem instances). In scenarios *SAT11-HAND*, *PROTEUS-2014* and *MAXSAT12-PMS*, for all the problem instances in the test set, the algorithms are sorted by their predicted performances. The average cumulative minimum of their actual performance in the sorted list is drawn at each TOPK elapsed step.

PMS, we visualize the cumulative minimum mean runtime for all approaches' predicting lists in Fig 5.6. On the left hand side, in scenario *SAT11-HAND*, though BLR (plotted with bold green yellow line) gives the worst recommendation at the $TOP1$ position, it reaches the optimal performance as *one hot random forest* does at position 4. Conversely, in scenario *PROTEUS-2014*, as plotted in the middle subplot, BLR finds the algorithm with shortest runtime at position 1 and beats all other approaches, while loses its dominant role gradually from position 3. Approaches like *single best*, *separated xgboost* and *separated random forest* take over the dominant positions from position 3. In scenario *MAXSAT12-PMS*, similar as in scenario *SAT11-HAND*, the recommendation from BLR reaches best at top position 3, in spite of the worst average run time of its predicted algorithms list at position 1. From this figure, we could see, when defining different $TOPK$ as the selection criteria, the ranking of the approaches finding the best algorithm can be totally different. Thus it makes sense trying different $TOPK$ selection criteria during the evaluation.

5.4.3 Accuracy and Inference Time Trade-off

As introduced in sub section 5.3.2, A3R is a good metric for measuring the combined effect of accuracy and time. We take this metric to make trade-off between accuracy and inference time for evaluating the AS approaches in this experiment. In order to make the accuracy/time ratio comparable across all scenarios, *one hot random forest regressor* (the approach wins on accuracy oriented metrics in most scenarios) is taken as reference approach (a_q) in the evaluation equation. It's drawn as the pink bar in the following figures, and the A3R value of this referred algorithm is always 1. All the accuracy metric values are calculated under $TOP3$ candidates selection.

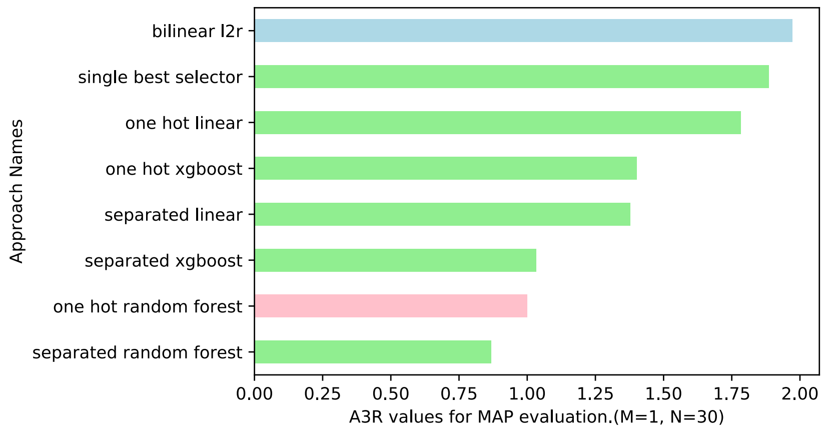


Figure 5.7: Approaches' Average A3R score across all the scenarios in terms of MAP and inference time. BLR performs the best when setting K with number 3 as the size of the candidates set.

As to precision oriented accuracy metrics (SUCC and MAP), the accuracy ratio is proportional to the metric value of the selected approach. Thus ACC value of a_q (referenced approach) is set as the denominator in the ratio formula $ACC^{s_i}_{a_p}/ACC^{s_i}_{a_q}$ in Eq. 5.22. Considering that inference time of different AS approaches span in 3 or 4 magnitudes, parameter for root N is set as 30 in the experiment to limit A3R in a reasonable range. As Fig. 5.7 shows, when evaluating the approaches regarding both MAP and inference time using A3R, BLR (in light blue bar) outperforms all other benchmark approaches, thus reaches the balance of model complexity and inference simplicity.

$$A3R(ACC)^{s_i}_{a_p a_q} = \frac{ACC^{s_i}_{a_p}/ACC^{s_i}_{a_q}}{\sqrt[N]{T^{s_i}_{a_p}/T^{s_i}_{a_q}}} \tag{5.22}$$

$$A3R(TC)^{s_i}_{a_p a_q} = \frac{\sqrt[M]{TC^{s_i}_{a_q}/TC^{s_i}_{a_p}}}{\sqrt[N]{T^{s_i}_{a_p}/T^{s_i}_{a_q}}} \tag{5.23}$$

For runtime oriented accuracy metrics (MCP and PAR10), their values are negatively correlated with prediction accuracy. The accuracy ratio $TC^{s_i}_{a_q}/TC^{s_i}_{a_p}$ therefore takes the metric value $TC^{s_i}_{a_p}$ as the denominator, like defined in Eq. 5.23. In addition, since the MCP and PAR10 metric value among approaches varies a lot even concerning magnitude, root parameter M is involved for this accuracy ratio as well to transform the ratio to a readable range. As Fig. 5.8 shows, for MCP, with the setting of $M = 3$ and $N = 30$, BLR (represented as light blue bar) again wins other benchmark approaches.

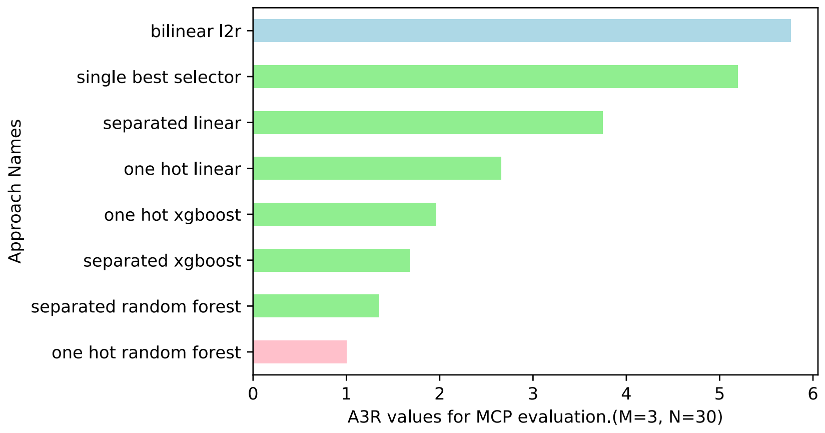


Figure 5.8: Approaches' Average A3R score across all the scenarios in terms of MCP and inference time. BLR performs the best when setting K with number 3 as the size of the candidates set.

5.4.4 Discussion

In this section, we observe the effect of BLR in AS scenarios. With concerns only on accuracy oriented approaches, BLR can be ranked in top3 approaches in some scenarios. When we propose applying A3R to balance the accuracy and inference time in the evaluation for AS, BLR is proven to be more effective across all the scenarios, especially when candidates selection criteria K is set to 3.

5.5 Top1 Bias and TopK Expansion

When considering the cumulative optimal algorithm in the $TOPK$ list, there are more chances to catch up the actual optimal algorithm for every AS approach. The AS effect will be different when choosing different K. In order to counter the **TOP1 evaluation bias** in the AS measurement, we propose expanding the selection criteria on the candidates set from $TOP1$ to $TOPK$. In this section, we study on the benefits and necessity of expanding the candidates set selection range from $TOP1$ to $TOPK$. Tentatively, we choose 1 and 3 as the default K selection.

5.5.1 Benefit of TOPK Selection

As discussed in the former sections, if we enlarge the algorithm candidates range from $TOP1$ to $TOPK$, we can expect the algorithms selected from a wider spectrum yielding different optimal selected algorithm. In this experiment, we tentatively set $K = 3$, and observe the difference on the cumulative evaluation result difference between the conditions

$K = 1$ and $K = 3$. For every AS scenario, we list the approach with the largest performance difference caused by $TOP1$ and $TOP3$ selection criteria, thus illustrate the benefit of the $TOPK$ expansion. We choose runtime oriented metrics MCP and PAR10 to represent the performance difference, considering their straightforward cumulative performance decrease along the $TOPK$ positions.

MCP calculates the runtime difference between the selected algorithm and the actual best algorithm. The lower MCP value, the better effect the AS approach possesses. Seen from Table 5.5, $TOP3$ selection criteria leads to the decreasing effect on MCP significantly compared to $TOP1$ criteria. We highlight the decrease percentage higher than 90.00% in red boxes in the table. The decrease percentage ranges from 55.78% to 100%. It demonstrates that enlarging the $TOPK$ candidates selection range can boost finding the algorithm *runtime* closer to the ground truth best.

The evaluation metric PAR10 gives 10 times penalty on the most recommended algorithm which is actually timeout. We also list the decrease percentage caused by $TOP3$ candidates expansion in Table 5.6. This decrease percentage falls in the interval 19.47% to 95.72%. The cases that the decrease percentage higher than 90.00% are highlighted in the red boxes in the table. This decreased percentage indicates the reduction of the possibility that the selected algorithm is timeout.

When expanding the $TOP1$ candidate set to the case $TOP3$, the observation of the significant decrease on the runtime metrics MCP and PAR10 confirms its benefit. In AS, under the parallel testing environment, the test on the $TOPK$ candidates stops at the *runtime* of the optimal algorithm in the candidates set. Thus the test time is also saved owing to the expansion. The selection of K depends on the computational power and environmental limit. Though $TOP1$ setting is required in most AS challenges, we suggest the expansion of this candidates selection range.

5.5.2 Necessity of Evaluation for different K choices

Expansion on $TOPK$ candidate selection has proven to bring performance benefit compared to only $TOP1$ selection (especially for run time consumption evaluation metrics). Here we study whether it's necessary to evaluate AS approaches for $TOP1$ and $TOPK$ candidates selection criteria separately. For every specific scenario, we use Kendall's Tau ranking correlation to measure the relevance of the AS approaches ranking under $TOP1$ and $TOP3$ selection criteria for each evaluation metric. As what is stated in Table 5.7, in most cases the correlations of ranking are weak between the two conditions, though there are several occasions with strong correlation (above 0.6) for the approaches ranking. The weak correlation indicates that there is no strong correlation between $Top1$ and $TopK$ evaluation results, the evaluation needs to be applied separately when a specific K is chosen as the candidate selection criteria.

Table 5.5: Improvement on MCP evaluation for Top3 and Top1 selection.

Scenario Name	Approach Name	MCP@1	MCP@3	MCP@3 - MCP@1	Difference in %
ASP-POTASSCO	One hot random forest	22.06	1.51	-20.55	-93.16%
BNSL-2016	Separated xgboost	217.94	27.88	-190.06	-87.21%
CPMP-2015	One hot random forest	120.11	0.41	-119.69	-99.66%
CSP-2010	Bilinear l2r	535.44	0.00	-535.44	-100.0%
CSP-MZN-2013	Separated random forest	68.78	8.36	-60.42	-87.84%
CSP-Minizinc-Obj-2016	Separated xgboost	216.13	95.57	-120.55	-55.78%
GRAPHS-2015	One hot xgboost	3774156.34	125048.31	-3649108.03	-96.69%
MAXSAT12-PMS	Separated random forest	34.24	0.04	-34.20	-99.88%
MAXSAT15-PMS-INDU	One hot random forest	77.08	5.47	-71.61	-92.9%
PROTEUS-2014	Single best selector	1023.12	39.61	-983.51	-96.13%
QBF-2011	One hot random forest	72.31	0.00	-72.31	-100.0%
QBF-2014	Separated xgboost	65.52	9.49	-56.03	-85.52%
SAT11-HAND	Bilinear l2r	1206.79	235.06	-971.74	-80.52%
SAT11-INDU	Separated random forest	582.31	75.73	-506.58	-86.99%
SAT11-RAND	Separated linear	456.10	16.44	-439.66	-96.4%
SAT12-ALL	Separated xgboost	213.24	59.09	-154.15	-72.29%
SAT12-INDU	One hot random forest	91.93	22.33	-69.59	-75.7%
SAT12-RAND	One hot random forest	69.13	11.11	-58.02	-83.93%
SAT15-INDU	Separated linear	689.23	148.96	-540.28	-78.39%
TSP-LION2015	Separated linear	76.95	4.24	-72.71	-94.5%

Table 5.6: Difference on PAR10 evaluation for Top3 and Top1 selection.

Scenario Name	Approach Name	PAR10@1	PAR10@3	PAR10@3 PAR10@1	Difference in %
ASP-POTASSCO	Bilinear l2r	1526.22	863.64	-662.58	-43.41%
BNSL-2016	Separated xgboost	2089.12	251.60	-1837.52	-87.96%
CPMP-2015	Separated random forest	5920.68	346.61	-5574.07	-94.15%
CSP-2010	Bilinear l2r	11032.73	6285.47	-4747.26	-43.03%
CSP-MZN-2013	Single best selector	9029.44	5538.58	-3490.86	-38.66%
CSP-Minizinc-Obj-2016	One hot random forest	2520.38	360.37	-2160.01	-85.7%
GRAPHS-2015	Separated linear	73415561.38	24868328.87	-48547232.51	-66.13%
MAXSAT12-PMS	Bilinear l2r	7056.63	5611.30	-1445.33	-20.48%
MAXSAT15-PMS-INDU	Bilinear l2r	4846.02	3018.59	-1827.43	-37.71%
PROTEUS-2014	Single best selector	9851.49	421.98	-9429.51	-95.72%
QBF-2011	Single best selector	15699.44	10371.48	-5327.96	-33.94%
QBF-2014	One hot xgboost	3696.26	2536.14	-1160.12	-31.39%
SAT11-HAND	Bilinear l2r	29375.15	19093.06	-10282.08	-35.0%
SAT11-INDU	One hot xgboost	17012.79	13700.37	-3312.42	-19.47%
SAT11-RAND	One hot linear	23445.32	9225.72	-14219.60	-60.65%
SAT12-ALL	Bilinear l2r	6318.53	2676.89	-3641.64	-57.63%
SAT12-INDU	One hot xgboost	3698.78	2656.96	-1041.82	-28.17%
SAT12-RAND	Separated xgboost	3843.91	3058.80	-785.11	-20.42%
SAT15-INDU	Separated linear	3143.63	443.36	-2700.28	-85.9%
TSP-LION2015	Separated random forest	255.34	25.21	-230.13	-90.13%

5.5.3 Discussion

The performance difference via all AS approaches proves that there is significant improvement on the AS effect when we altering the candidates range from $TOP1$ to $TOPK$, especially for *runtime* relevant evaluation metrics. Since approaches' rankings are weakly correlated for different K setting selection criteria in most conditions, evaluating AS approaches for specific k is necessary. Therefore, the research problem **RP5.3 Whether we can expand the selection range thus benefit from the cumulative optimal algorithm?** is solved in this section.

5.6 Conclusions

In this chapter, we build up the research towards the evaluation bias in the algorithm recommendation scenario. We propose Bi-Linear Learning to Rank (BLR) to solve AS problem with the consideration of the balance between accuracy and inference time. With the list-wise probability assumption, it models the uncertainty in the algorithm performance. The learning process of BLR averts the problems like multi-models training and algorithms' one-hot conversion in traditional AS benchmark approaches. To counter the limit of accuracy oriented evaluation bias, inference time is also involved in the evaluation protocol. A3R is chosen to balance the evaluation bias. When choosing 3 as the candidates set selection criteria, BLR outweighs other AS approaches regarding A3R. Finally, we affirm the benefit and meaning of expanding the selection range of candidate approaches from $TOP1$ to $TOPK$ regarding the cumulative optimal demand of AS evaluation, and propose to use it in the AS scenarios.

Table 5.7: Correlation for approaches ranking between top3 and top1 candidate selection.

Scenario Name	Correlation TOP1/TOP3 SUCC	Correlation TOP1/TOP3 MAP	Correlation TOP1/TOP3 MCP	Correlation TOP1/TOP3 PAR10
ASP-POTASSCO	0.500	0.611	0.833	0.500
BNSL-2016	-0.111	0.389	-0.556	-0.556
CPMP-2015	0.111	0.111	0.000	-0.056
CSP-2010	-0.222	-0.222	0.278	0.444
CSP-MZN-2013	0.556	-0.056	0.556	0.833
CSP-Minizinc-Obj-2016	0.278	-0.444	0.167	0.056
GRAPHS-2015	0.444	0.222	0.389	-0.056
MAXSAT12-PMS	0.167	0.222	0.056	0.111
MAXSAT15-PMS-INDU	0.556	-0.389	0.111	0.056
PROTEUS-2014	0.556	0.222	-0.333	0.278
QBF-2011	0.944	0.056	0.389	0.556
QBF-2014	0.611	0.833	0.944	0.389
SAT11-HAND	0.389	0.278	0.111	0.444
SAT11-INDU	0.889	0.722	0.444	0.056
SAT11-RAND	0.167	-0.167	-0.056	-0.333
SAT12-ALL	0.778	-0.056	0.111	0.833
SAT12-INDU	0.556	0.944	0.889	0.556
SAT12-RAND	0.389	0.278	0.444	0.222
SAT15-INDU	0.333	0.111	0.333	0.278
TSP-LION2015	0.556	0.278	0.556	0.667

CHAPTER 6

CONCLUSION AND OUTLOOK

This thesis mainly focuses on representing, analyzing and balancing the bias in recommender systems. Given the characteristics in the recommender scenarios such as IPTV system, digital news portal and algorithm selection, we target the main research problems and contributions accordingly. In this chapter, we revisit the research problems and the contributions in each scenario and look into the potential extension of the current work.

6.1 Conclusions

In RS, the biased data or user habit easily leads to the *filtering bubble* and make users limited in the inherent area. The bias phenomenon can be understood from different perspectives like selection bias, presentation bias, dataset bias, demographic bias, evaluation bias etc. In this thesis, we mainly solve three research problems: 1) contextual bias in IPTV scenario, 2) content level bias understanding in the digital news portals, and 3) trade off on evaluation bias in algorithm selection cases. Chapter 1 introduces the main research goal and the general contributions. Thereafter, the relevant background and related works have been summarized in Chapter 2 regarding each scenario we are dealing with. More specifically, Chapter 3 to Chapter 5 illustrate the concrete solutions we proposed to deal with the concerned biases conditions.

Chapter 3 investigates first research problem of the contextual bias in IPTV scenario. The user study was conducted to dig into users' opinion on the influence of the context factors on their choices in IPTV system. In this user study, we also figured out that users' individual inclination between personal habit and contextual influence varies. Therefore, when proposing the context-aware recommendation model, we expand the LDA to the context-aware format which involving user's preference selection path in the probabilistic graph to represent user's inclination. The proposed model reaches the balance on the accuracy and diversity recommendation effect measurement. Nevertheless, the model training based on the system internal data still limit the learning scope to users' historical selection. To solve this issue, the domain relevant event trends and peaks from the external social has been proposed to counter the bias raised from the internal system. The evaluation on the contextual grid also support the effect of the recommendation content balancing.

In Chapter 4, through the participation of news recommendation challenge NewsREEL, we found out that popularity based approaches always outweigh other approaches in the board. The analysis via impression or click granularity trace the the bias on the article granularity. Nevertheless, the lack of the understanding from the content level of the bias makes the popularity a black box for the users. Therefore, we extract the popularity from the topic level, view such topic level bias from both temporal and user behavior views. Such decomposition of the bias understanding of news depicts the clear picture that the bias on news are actually dominated by the breaking events. These breaking events can be influential to multiple articles and lead to the popularity bias. Therefore, we propose to parse the topics from both articles and content level. The TCD and TRB are defined to represent the bias degree for terms and ranked lists separately. The analysis from both temporal and behavioral view presents the correlation between popularity bias and trend prediction power thoroughly on the four digital news portals.

Chapter 5 turns into the recommendation scenario without real users, e.g. recommending algorithms to the unseen problem instances in the algorithm selection tasks. In this case, the bias problems are not attributed to users' behavior or dataset acquiring, since the performances of the algorithms on the problem instances are more determined. The minor uncertainty in this scenario is mostly resulted from either the computational sources difference or the sampling approaches variance. When talking about the bias phenomenon in algorithm recommendation, the evaluation bias becomes the main concern. Over reliance on the accuracy oriented measurements and the limit on the $TOP1$ predicted item leads to the evaluation bias in the algorithm selection problems. The Bi-Linear Learning to Rank (BLR) is proposed in this chapter to learn and predict the ranking of algorithms regarding a new problem instance. BLR escapes the extra steps in the conventional algorithm selection approaches, i.e. intermediate one-hot encoding and separates training models building. As the proposed evaluation metric, A3R realizes the trade off between the accuracy and inference time considerations. BLR has proven to outperform other traditional approaches regarding this balanced evaluation metric. Such advantage is reflected particularly when expanding the recommendation candidates from $TOP1$ to $TOPK$. The expansion of the algorithms candidates from $TOP1$ to $TOPK$ won't raise any extra computational resources, but brings in much benefits in the selection effect. Therefore, we propose to consider such $TOPK$ expansion during the evaluation to get rid of the bias caused by the $TOP1$ limitation.

6.2 Outlook and Future Work

This section discuss the potential future work of the research in this thesis. In a general view, out of the scope of the types of bias problems researched in this thesis (countering contextual bias, topic level understanding of the bias, and the evaluation bias in the typical recommendation scenarios), there are many other challenges of bias problems in RS worthy of study. For instance, the presentation bias and ranking bias are the typical bias phenomenon regarding the user behavior. They mainly appear due to the users' habitual interaction with the front end presentation of the system, thus user's true preference ranking needs to be corrected based on the ranking order they are presented to the user. In addition, the aggregated data bias resulted from the algorithm bias makes the algorithm selection in the RS prone to choose the algorithms similar to the deployed algorithms in the system. Though there have been research conducted towards these extended bias problems, it still makes sense borrowing the ideas from this thesis to solve them in a brand new way. For instance, introducing external sources to help the dataset acquisition get rid of the algorithm bias.

In line with the specific bias research problem in the thesis, there are also corresponding future work. First, from the view of countering contextual bias, though the context-aware LDA creates the possibility of capturing users' inclination on the contextual factor and the self preference, the contextual adaptation is still across the whole system, yet not individually distinguished. The model incorporating individual contextual preferences also deserves further study. Aside from that, when involving the external sources into the internal system to counter the bias, the *trends* or *peaks* tracked from the system is on the granularity of specific TV programs. However, there are plenty of meta features of a TV program, e.g. genre, region, directors, actors, etc.. The *trend* or *peaks* we aggregate from the external source can be expanded to these extra fields in the future.

Additionally, as to the content level understanding of bias in the digital news portal, the concern degree and the topic ranking bias are defined on the term level in the thesis. Though the terms are all domain relevant, it's possible the entity level information is missing. Therefore, the extra named entity recognition phase can be treated as the next step of the research work. On the other hand, the observations of term ranking bias from both behavioral view and temporal view can be considered as recommendation indicators. The prediction power of the term ranking bias is different for specific categories of news portals. It makes sense to make use of such ranking bias to boost other relevant news in the long tail yet still relevant to the popularity on the content level.

As for the approach we proposed against evaluation bias in the algorithm selection, there are also potential future directions for the research. Regarding the algorithm BLR, since it's a model with non-convex loss definition, the convergence criteria can be adjusted to tune better parameters setting and reach the better algorithm selection effect. When thinking about the algorithm selection scenarios, we only investigate 20 AS scenarios in this thesis. Extending the experiments to more additional scenarios will give a stronger confidence on the evaluation result. For illustrating the effect of the $TOPK$ expansion, we only deal with the expansion with K as 3 during the experiment phase. Yet more thorough study can be done on the selection criteria of K to meet the balance of performance gain and computational power.

6.3 Final Remarks

The bias phenomenon in RS system can be resulted from different reasons in a specific recommendation scenario. Considering the characteristics in each specific RS scenario, the algorithm model and bias processing procedure need to be customized to meet the special need. In the scenarios that contextual patterns are obvious, like IPTV system, we put more effort on the contextual bias detection and countering. While for the literal plentiful recommendation scenario like digital news portals, tracking the ranking trend from the topic level provides the explainable understanding of the users' biased choices. The fine-grained biased topics can serve as the understandable indicator while making further recommendation for the users. When coming into the algorithm selection scenarios, where data set are less bias prone, the evaluation bias attracts more attention. The multi-objective evaluation metrics and $TOPK$ expansion are the suggested methods for the bias trade off in the evaluation process. Along with the change of the evaluation metric and the optimization target, the focus of the algorithm design and modeling need to take the factors from all objectives in the evaluation into consideration. In this thesis, we propose the concrete approaches and solutions concerning all the special bias conditions in the recommendation scenarios mentioned above and verify their advantages and effects. For the future works, they can be conducted from the views of either expanding the scope of the bias or more comprehensive modeling or feature tuning.

BIBLIOGRAPHY

[1] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender Systems: An Introduction*. Cambridge University Press, New York, NY, USA, 1st edition, 2010. (Cited on pages 1, 10 and 16.)

[2] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor. *Recommender Systems Handbook*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010. (Cited on pages 1 and 16.)

[3] Jie Lu, Dianshuang Wu, Mingsong Mao, Wei Wang, and Guangquan Zhang. Recommender system application developments: A survey. *Decision Support Systems*, 74(Supplement C):12 – 32, 2015. (Cited on page 1.)

[4] Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender systems. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, RecSys '08, pages 335–336, New York, NY, USA, 2008. ACM. (Cited on pages 2, 10 and 11.)

[5] Gediminas Adomavicius, Jesse Bockstedt, Curley Shawn, and Jingjing Zhang. *Debiasing user preference ratings in recommender systems*, volume 1253, pages 2–9. CEUR-WS, 2014. (Cited on pages 2, 15 and 29.)

[6] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. Controlling popularity bias in learning-to-rank recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, RecSys '17, pages 42–46, New York, NY, USA, 2017. ACM. (Cited on pages 2, 6, 12, 19, 21 and 29.)

[7] Sushma Channamsetty and Michael D. Ekstrand. Recommender response to diversity and popularity bias in user profiles. In *FLAIRS Conference*, 2017. (Cited on page 2.)

[8] Arnaud De Myttenaere, Bénédicte Le Grand, Boris Golden, and Fabrice Rossi. Reducing offline evaluation bias in recommendation systems. *CoRR*, abs/1407.0822, 2014. (Cited on page 2.)

[9] Michael D. Ekstrand, Mucun Tian, Ion Madrazo Azpiazu, Jennifer D. Ekstrand, Oghenemaro Anuyah, David McNeill, and Maria Soledad Pera. All the cool kids, how do they fit in?: Popularity and demographic biases in recommender evaluation and effectiveness. In Sorelle A. Friedler and Christo Wilson, editors, *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, volume 81 of *Proceedings of Machine Learning Research*, pages 172–186, New York, NY, USA, 23–24 Feb 2018. PMLR. (Cited on page 2.)

[10] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. An experimental comparison of click position-bias models. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, WSDM '08, pages 87–94, New York, NY, USA, 2008. ACM. (Cited on pages 2 and 15.)

[11] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. Unbiased learning-to-rank with biased feedback. *CoRR*, abs/1608.04468, 2016. (Cited on pages 2 and 15.)

[12] Alejandro Bellogín, Pablo Castells, and Iván Cantador. Statistical biases in information retrieval metrics for recommender systems. *Information Retrieval Journal*, 20:1–29, 07 2017. (Cited on page 3.)

[13] Jing Yuan, Fikret Sivrikaya, Stefan Marx, and Frank Hopfgartner. When to recommend what? a study on the role of contextual factors in ip-based tv services. In *MindTheGap@iConference*, pages 12–18, 2014. (Cited on pages 5 and 42.)

[14] Jing Yuan. How adaptive aggregate contextual recommender benefits ip-based tv services. 06 2014. (Cited on page 5.)

[15] Jing Yuan, Fikret Sivrikaya, Frank Hopfgartner, Andreas Lommatzsch, and Mu Mu. Context-aware LDA: Balancing Relevance and Diversity in TV Content Recommenders. In *Proceedings of the 2nd Workshop on Recommendation Systems for Television and Online Videos (RecSysTV 2015)*, September 2015. (Cited on pages 5, 27 and 28.)

[16] Mu Mu Jing Yuan, Andreas Lommatzsch. Applying topic model in context-aware tv programs recommendation. In *Proc. of the LWDA conference, Track Knowledge Manage*, page 52. CEUR Workshop Proceedings, 2016. (Cited on page 5.)

[17] Felix Lorenz, Jing Yuan, Andreas Lommatzsch, Mu Mu, Nicholas Race, Frank Hopfgartner, and Sahin Albayrak. Countering contextual bias in tv watching behavior: Introducing social trend as external contextual factor in tv recommenders.

In *Proceedings of the 2017 ACM International Conference on Interactive Experiences for TV and Online Video*, TVX '17, pages 21–30, New York, NY, USA, 2017. ACM. (Cited on page 5.)

[18] Jing Yuan, Andreas Lommatzsch, and Benjamin Kille. Clicks pattern analysis for online news recommendation systems. In *Working Notes of CLEF 2016 - Conference and Labs of the Evaluation forum, Évora, Portugal, 5-8 September, 2016.*, pages 679–690, 2016. (Cited on pages 5, 73 and 83.)

[19] Jing Yuan, Christian Geissler, Weijia Shao, Andreas Lommatzsch, and Brijnesh Jain. When algorithm selection meets bi-linear learning to rank: Accuracy and inference time trade off with candidates expansion. *International Journal of Data Science and Analytics*, 2020. (Cited on page 5.)

[20] Paolo Cremonesi and Roberto Turrin. Analysis of cold-start recommendations in iptv systems. In Lawrence D. Bergman, Alexander Tuzhilin, Robin D. Burke, Alexander Felfernig, and Lars Schmidt-Thieme, editors, *RecSys*, pages 233–236. ACM, 2009. (Cited on page 9.)

[21] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pages 263–272, Washington, DC, USA, 2008. IEEE Computer Society. (Cited on pages 9 and 10.)

[22] Paolo Cremonesi, Roberto Turrin, and Fabio Airoldi. Hybrid algorithms for recommending new items. In *Proceedings of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, HetRec '11, pages 33–40, New York, NY, USA, 2011. ACM. (Cited on pages 9, 22 and 23.)

[23] Tong Queue Lee, Young Park, and Yong-Tae Park. A time-based approach to effective recommender systems using implicit feedback. *Expert Systems with Applications*, 34(4):3055 – 3062, 2008. (Cited on page 9.)

[24] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734 – 749, june 2005. (Cited on page 10.)

[25] Greg Linden, B. Smith, and J. York. Amazon.com recommendations: item-to-item collaborative filtering. *Internet Computing, IEEE*, 7:76 – 80, 02 2003. (Cited on page 10.)

[26] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 79–86. ACM, 2010. (Cited on pages 10, 12, 27 and 28.)

[27] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009. (Cited on pages 10, 11 and 22.)

[28] Tianqi Chen, Weinan Zhang, Qiuxia lu, Kailong Chen, Zhao Zheng, and Yong Yu. Svdfeature: A toolkit for feature-based collaborative filtering. *The Journal of Machine Learning Research*, 13:3619–3622, 12 2012. (Cited on page 10.)

[29] Balaji Lakshminarayanan, Guillaume Bouchard, and Cedric Archambeau. Robust bayesian matrix factorisation. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 425–433, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. (Cited on page 10.)

[30] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John T. Riedl. Application of dimensionality reduction in recommender system – a case study. In *IN ACM WEBKDD WORKSHOP*, 2000. (Cited on page 10.)

[31] Yehuda Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’08, pages 426–434, New York, NY, USA, 2008. ACM. (Cited on page 11.)

[32] Yehuda Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’09, pages 447–456, New York, NY, USA, 2009. ACM. (Cited on pages 11, 15, 22 and 28.)

[33] Paolo Cremonesi and Roberto Turrin. Time-evolution of iptv recommender systems. In *Proceedings of the 8th European Conference on Interactive TV and Video*, EuroITV ’10, pages 105–114, New York, NY, USA, 2010. ACM. (Cited on pages 11 and 22.)

[34] Neal Lathia, Stephen Hailes, Licia Capra, and Xavier Amatriain. Temporal diversity in recommender systems. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ’10, pages 210–217, New York, NY, USA, 2010. ACM. (Cited on pages 11, 28 and 47.)

[35] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, and Alan Hanjalic. Cars2: Learning context-aware representations for context-aware recommendations. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, pages 291–300. Association for Computing Machinery, 2014. (Cited on page 12.)

[36] Pedro G Campos, Fernando Díez, and Iván 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. (Cited on pages 12 and 28.)

[37] Hongzhi Yin, Bin Cui, Ling Chen, Zhiting Hu, and Zi Huang. A temporal context-aware model for user behavior modeling in social media systems. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 1543–1554. ACM, 2014. (Cited on pages 12 and 28.)

[38] Xuanhui Wang, Michael Bendersky, Donald Metzler, and Marc Najork. Learning to rank with selection bias in personal search. In *Proc. of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 115–124, 2016. (Cited on pages 12, 15 and 21.)

[39] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. Unbiased learning-to-rank with biased feedback. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, WSDM '17, pages 781–789, New York, NY, USA, 2017. ACM. (Cited on pages 12, 15 and 21.)

[40] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: From pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 129–136, New York, NY, USA, 2007. ACM. (Cited on pages 12, 21 and 108.)

[41] Shoujin Wang, Longbing Cao, and Yan Wang. A survey on session-based recommender systems. *CoRR*, abs/1902.04864, 2019. (Cited on page 12.)

[42] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2Nd ACM Conference on Electronic Commerce*, EC '00, pages 158–167, New York, NY, USA, 2000. ACM. (Cited on page 12.)

[43] Timur Osadchiy, Ivan Poliakov, Patrick Olivier, Maisie Rowland, and Emma Foster. Recommender system based on pairwise association rules. *Expert Systems with Applications*, 115:535 – 542, 2019. (Cited on page 12.)

[44] Jangwon Gim, Hanmin Jung, and Do-Heon Jeong. Xonto-apriori: An effective association rule mining algorithm for personalized recommendation systems. In James J. (Jong Hyuk) Park, Ivan Stojmenovic, Hwa Young Jeong, and Gangman Yi, editors, *Computer Science and its Applications*, pages 1131–1138, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. (Cited on page 12.)

[45] V. Swathi and Madadi Sunitha Reddy. Music recommendation system using association rules. *International Journal of Technology Enhancements and Emerging Engineering Research*, 2:31–34, 2014. (Cited on page 12.)

[46] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, and Dasarathi Sampath. The youtube video recommendation system. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 293–296, New York, NY, USA, 2010. ACM. (Cited on pages 13 and 24.)

[47] Francesco Bonchi, Raffaele Perego, Fabrizio Silvestri, Hossein Vahabi, and Rossano Venturini. Recommendations for the long tail by term-query graph. pages 15–16, 01 2011. (Cited on page 13.)

[48] Shuo Chen, Joshua L. Moore, Douglas Turnbull, and Thorsten Joachims. Playlist prediction via metric embedding. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 714–722. Association for Computing Machinery, 2012. (Cited on page 14.)

[49] Liang Hu, Longbing Cao, Shoujin Wang, Guandong Xu, Jian Cao, and Zhiping Gu. Diversifying personalized recommendation with user-session context. In Carles Sierra, editor, *IJCAI*, pages 1858–1864. ijcai.org, 2017. (Cited on page 14.)

[50] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *Computing Research Repository*, abs/1511.06939, 2016. (Cited on page 14.)

[51] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar, October 2014. Association for Computational Linguistics. (Cited on page 14.)

[52] Balázs Hidasi and Alexandros Karatzoglou. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM 18, pages

843–852, New York, NY, USA, 2018. Association for Computing Machinery. (Cited on page 14.)

[53] Malte Ludewig and Dietmar Jannach. Evaluation of session-based recommendation algorithms. *CoRR*, abs/1803.09587, 2018. (Cited on page 15.)

[54] Mahmoudreza Babaei, Abhijnan Chakraborty, Juhi Kulshrestha, Elissa M. Redmiles, Meeyoung Cha, and Krishna P. Gummadi. Analyzing biases in perception of truth in news stories and their implications for fact checking. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, page 139, New York, NY, USA, 2019. Association for Computing Machinery. (Cited on page 15.)

[55] Zachary A. Pardos and Weijie Jiang. Combating the filter bubble: Designing for serendipity in a university course recommendation system. *CoRR*, abs/1907.01591, 2019. (Cited on page 15.)

[56] Denis Kotkov, Shuaiqiang Wang, and Jari Veijalainen. A survey of serendipity in recommender systems. *Knowledge-Based Systems*, pages 180–192, 2016. (Cited on page 15.)

[57] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. Recommendations as treatments: Debiasing learning and evaluation. *CoRR*, abs/1602.05352, 2016. (Cited on page 15.)

[58] Li Chen, Yonghua Yang, Ningxia Wang, Keping Yang, and Quan Yuan. How serendipity improves user satisfaction with recommendations? a large-scale user evaluation. In *The World Wide Web Conference*, WWW '19, pages 240–250, New York, NY, USA, 2019. Association for Computing Machinery. (Cited on page 15.)

[59] Yisong Yue, Rajan Patel, and Hein Roehrig. Beyond position bias: Examining result attractiveness as a source of presentation bias in clickthrough data. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 1011–1018, New York, NY, USA, 2010. ACM. (Cited on page 15.)

[60] Zhichong Fang, Aman Agarwal, and Thorsten Joachims. Intervention harvesting for context-dependent examination-bias estimation. *CoRR*, abs/1811.01802, 2018. (Cited on page 15.)

[61] Aman Agarwal, Ivan Zaitsev, Xuanhui Wang, Cheng Li, Marc Najork, and Thorsten Joachims. Estimating position bias without intrusive interventions. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, WSDM '19, pages 474–482, New York, NY, USA, 2019. Association for Computing Machinery. (Cited on page 15.)

[62] Alan Said, Ben Fields, Brijnesh J. Jain, and Sahin Albayrak. User-centric evaluation of a k-furthest neighbor collaborative filtering recommender algorithm. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, pages 1399–1408, New York, NY, USA, 2013. Association for Computing Machinery. (Cited on page 15.)

[63] Alan Menk, Laura Sebastia, and Rebeca Ferreira. Curumim: A serendipitous recommender system for tourism based on human curiosity. 11 2017. (Cited on page 15.)

[64] Valentina Maccatrozzo. *Burst the Filter Bubble: Using Semantic Web to Enable Serendipity*, pages 391–398. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. (Cited on page 15.)

[65] Titipat Achakulvisut, Daniel Acuna, Tulakan Ruangrong, and Konrad Kording. Science concierge: A fast content-based recommendation system for scientific publications. *PLOS ONE*, 11, 04 2016. (Cited on page 16.)

[66] Riccardo Bambini, Paolo Cremonesi, and Roberto Turrin. A recommender system for an iptv service provider: a real large-scale production environment. In *Recommender Systems Handbook*, pages 299–331. Springer US, 2011. (Cited on page 16.)

[67] Jing Yuan, Quan Zheng, Zhijun Sun, and Song Wang. Research on the technology of video semantic retrieval based on structured semantic strings. In Yinglin Wang and Tianrui Li, editors, *Foundations of Intelligent Systems*, pages 721–730, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. (Cited on page 17.)

[68] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE*, 41(6):391–407, 1990. (Cited on page 17.)

[69] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 50–57, New York, NY, USA, 1999. ACM. (Cited on page 18.)

[70] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003. (Cited on pages 18 and 40.)

[71] Daniel Soutner and Luděk Müller. Application of lstm neural networks in language modelling. In Ivan Habernal and Václav Matoušek, editors, *Text, Speech, and Dialogue*, pages 105–112, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. (Cited on page 18.)

[72] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. (Cited on page 18.)

[73] Xiangyu Zhao, Zhendong Niu, and Wei Chen. Opinion-based collaborative filtering to solve popularity bias in recommender systems. In Hendrik Decker, Lenka Lhotská, Sebastian Link, Josef Basl, and A. Min Tjoa, editors, *Database and Expert Systems Applications*, pages 426–433, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. (Cited on page 19.)

[74] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. Managing popularity bias in recommender systems with personalized re-ranking. *CoRR*, abs/1901.07555, 2019. (Cited on page 19.)

[75] Sorelle A. Friedler, Carlos Scheidegger, Suresh Venkatasubramanian, Sonam Choudhary, Evan P. Hamilton, and Derek Roth. A comparative study of fairness enhancing interventions in machine learning. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 329–338, New York, NY, USA, 2019. Association for Computing Machinery. (Cited on page 19.)

[76] P. Vijaya Kumar and V. Raghunatha Reddy. A survey on recommender systems (rss) andits applications. *International Journal of Innovative Research in Computer and Communication Engineering*, 2:5254–5260, 2014. (Cited on page 20.)

[77] Mustafa Misir and Michèle Sebag. Algorithm selection as a collaborative filtering problem. Research report, December 2013. (Cited on pages 20, 21 and 98.)

[78] Ilya Makarov, Oleg Bulanov, and Leonid E. Zhukov. Co-author recommender system. In Valery A. Kalyagin, Alexey I. Nikolaev, Panos M. Pardalos, and Oleg A. Prokopyev, editors, *Models, Algorithms, and Technologies for Network Analysis*, pages 251–257, Cham, 2017. Springer International Publishing. (Cited on page 20.)

[79] Bernd Bischl, Pascal Kerschke, Lars Kotthoff, Marius Thomas Lindauer, Yuri Malitsky, Alexandre Fréchette, Holger H. Hoos, Frank Hutter, Kevin Leyton-Brown, Kevin Tierney, and Joaquin Vanschoren. Aslib: A benchmark library for algorithm selection. *CoRR*, abs/1506.02465, 2015. (Cited on pages 20 and 100.)

[80] Lars Kotthoff, Barry Hurley, and Barry O'Sullivan. The ICON challenge on algorithm selection. *AI Magazine*, 38(2):91–93, 2017. (Cited on pages 20, 25 and 100.)

[81] Marius Lindauer, Frank Hutter, Holger H. Hoos, and Torsten Schaub. Autofolio: An automatically configured algorithm selector (extended abstract). In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI*

2017, Melbourne, Australia, August 19-25, 2017, pages 5025–5029, 2017. (Cited on pages 20 and 98.)

[82] John R. Rice. The algorithm selection problem. volume 15 of *Advances in Computers*, pages 65 – 118. Elsevier, 1976. (Cited on page 20.)

[83] Troy A. Johnson and Rudolf Eigenmann. Context-sensitive domain-independent algorithm composition and selection. In *Proceedings of the 27th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '06, pages 181–192, New York, NY, USA, 2006. ACM. (Cited on page 20.)

[84] Frank Hutter, Lin Xu, Holger H. Hoos, and Kevin Leyton-Brown. Algorithm runtime prediction: Methods and evaluation. *Artificial Intelligence*, 206:79 – 111, 2014. (Cited on pages 20 and 97.)

[85] Pascal Kerschke, Holger H. Hoos, Frank Neumann, and Heike Trautmann. Automated algorithm selection: Survey and perspectives. *CoRR*, abs/1811.11597, 2018. (Cited on page 20.)

[86] Mao Luo, Chu-Min Li, Fan Xiao, Felip Manya, and Zhipeng Lv. An effective learnt clause minimization approach for cdcl sat solvers. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 703–711, 2017. (Cited on pages 20 and 25.)

[87] Jie-Hong R. Jiang Nian-Ze Lee, Yen-Shi Wang. Solving stochastic boolean satisfiability under random-exist quantification. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 688–694, 2017. (Cited on pages 20 and 25.)

[88] Andreia Mordido Carlos Caleiro, Filipe Casal. Classical generalized probabilistic satisfiability. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 908–914, 2017. (Cited on pages 20 and 25.)

[89] Fahiem Bacchus, Antti Hyttinen, Matti Järvisalo, and Paul Saikko. Reduced cost fixing in maxsat. In *Principles and Practice of Constraint Programming - 23rd International Conference, CP 2017, Melbourne, VIC, Australia, August 28 - September 1, 2017, Proceedings*, pages 641–651, 2017. (Cited on pages 20 and 25.)

[90] Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Satzilla: Portfolio-based algorithm selection for sat. *J. Artif. Int. Res.*, 32(1):565–606, June 2008. (Cited on page 20.)

[91] Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Satzilla: Portfolio-based algorithm selection for SAT. *CoRR*, abs/1111.2249, 2011. (Cited on pages 20 and 98.)

[92] Marius Thomas Lindauer, Holger H. Hoos, Frank Hutter, and Torsten Schaub. Autofolio: An automatically configured algorithm selector. *J. Artif. Intell. Res.*, 53:745–778, 2015. (Cited on page 20.)

[93] L. Kotthoff, Colleen Thornton, Holger Hoos, F. Hutter, and Kevin Leyton-Brown. Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka. *Journal of Machine Learning Research*, 18:1–5, 03 2017. (Cited on page 21.)

[94] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2962–2970. Curran Associates, Inc., 2015. (Cited on page 21.)

[95] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Tobias Springenberg, Manuel Blum, and Frank Hutter. *Auto-sklearn: Efficient and Robust Automated Machine Learning*, pages 113–134. Springer International Publishing, Cham, 2019. (Cited on page 21.)

[96] David H. Stern, Ralf Herbrich, and Thore Graepel. Matchbox: Large scale online bayesian recommendations. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 111–120, New York, NY, USA, 2009. ACM. (Cited on page 21.)

[97] David Stern, Ralf Herbrich, Thore Graepel, Horst Samulowitz, Luca Pulina, and Armando Tacchella. Collaborative expert portfolio management. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI'10, pages 179–184. AAAI Press, 2010. (Cited on page 21.)

[98] Mustafa Mısır and Michèle Sebag. Alors: An algorithm recommender system. *Artificial Intelligence*, 244:291–314, 2017. (Cited on page 21.)

[99] Chengrun Yang, Yuji Akimoto, Dae Won Kim, and Madeleine Udell. Oboe: Collaborative filtering for automl model selection. In *Proceedings of the Twenty-Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1173–1183. Association for Computing Machinery, 2019. (Cited on page 21.)

[100] Richard J. Oentaryo, Stephanus Daniel Handoko, and Hoong Chuin Lau. Algorithm selection via ranking. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 1826–1832. AAAI Press, 2015. (Cited on page 21.)

[101] Fálix Hernández del Olmo and Elena Gaudioso. Evaluation of recommender systems: A new approach. *Expert Syst. Appl.*, 35:790–804, 10 2008. (Cited on page 21.)

[102] Yi Ding and Xue Li. Time weight collaborative filtering. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, CIKM '05, pages 485–492, New York, NY, USA, 2005. ACM. (Cited on page 22.)

[103] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 39–46, New York, NY, USA, 2010. ACM. (Cited on page 22.)

[104] Yu-Cheng Chen, Hsien-Chao Huang, and Yueh-Min Huang. Community-based program recommendation for the next generation electronic program guide. *Consumer Electronics, IEEE Transactions on*, 55(2):707 –712, may 2009. (Cited on page 22.)

[105] Yu Xin and Harald Steck. Multi-value probabilistic matrix factorization for ip-tv recommendations. In *Proceedings of the fifth ACM conference on Recommender systems*, RecSys '11, pages 221–228, New York, NY, USA, 2011. ACM. (Cited on page 22.)

[106] Andreas Lommatzsch, Till Plumbaum, and Sahin Albayrak. A linked dataverse knows better: Boosting recommendation quality using semantic knowledge. In *Proc. of the 5th Intl. Conf. on Advances in Semantic Processing*, pages 97 – 103, Wilmington, DE, USA, 2011. IARIA. (Cited on page 22.)

[107] Mario Rodriguez, Christian Posse, and Ethan Zhang. Multiple objective optimization in recommender systems. In *Proceedings of the sixth ACM conference on Recommender systems*, RecSys '12, pages 11–18, New York, NY, USA, 2012. ACM. (Cited on pages 22 and 24.)

[108] Benjamin Kille, Andreas Lommatzsch, Gebrekirstos G. Gebremeskel, Frank Hopfgartner, Martha Larson, Jonas Seiler, Davide Malagoli, András Serény, Torben Brodt, and Arjen P. de Vries. Overview of newsreel'16: Multi-dimensional evaluation of real-time stream-recommendation algorithms. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction - 7th International Conference of the CLEF Association, CLEF 2016, Évora, Portugal, September 5-8, 2016, Proceedings*, pages 311–331, 2016. (Cited on pages 24 and 72.)

[109] S. Song, H. Moustafa, and H. Afifi. Advanced iptv services personalization through context-aware content recommendation. *IEEE Transactions on Multimedia*, 14(6):1528–1537, Dec 2012. (Cited on page 24.)

[110] Carla P. Gomes and Bart Selman. Algorithm portfolios. *Artificial Intelligence*, 126(1):43 – 62, 2001. Tradeoffs under Bounded Resources. (Cited on page 25.)

[111] Carla P. Gomes and Bart Selman. Algorithm portfolio design: Theory vs. practice. *CoRR*, abs/1302.1541, 2013. (Cited on page 25.)

[112] Kevin Leyton-Brown, Eugene Nudelman, Galen Andrew, Jim McFadden, and Yoav Shoham. A portfolio approach to algorithm select. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, IJCAI'03, pages 1542–1543, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc. (Cited on page 25.)

[113] Balázs Hidasi and Domonkos Tikk. General factorization framework for context-aware recommendations. *Data Mining and Knowledge Discovery*, 30(2):342–371, 2016. (Cited on pages 27 and 28.)

[114] Marko Krstic and Milan Bjelica. Context-aware personalized program guide based on neural network. *IEEE Trans. on Consumer Electronics*, 58(4):1301–1306, 2012. (Cited on pages 27 and 28.)

[115] Ericsson ConsumerLab. Tv and media 2015. Technical report, Ericsson ConsumerLab, 2015. (Cited on page 28.)

[116] Ericsson ConsumerLab. Tv and media 2016. Technical report, Ericsson ConsumerLab, 2016. (Cited on page 28.)

[117] Hong-Yi Chang, Shih-Chang Huang, and Chih-Chun Lai. A personalized iptv channel-recommendation mechanism based on the mapreduce framework. *The Journal of Supercomputing*, 69(1):225–247, Jul 2014. (Cited on page 28.)

[118] Jorge Abreu, Pedro Almeida, Bruno Teles, and Márcio Reis. Viewer behaviors and practices in the (new) television environment. In *Procs. of the 11th european conf. on Interactive TV and video*, pages 5–12. ACM, 2013. (Cited on page 28.)

[119] Fábio Santos da Silva, Luiz Gustavo Pacola Alves, and Graça Bressan. Personaltvware: A proposal of architecture to support the context-aware personalized recommendation of tv programs. In *European Interactive TV Conference (EuroITV 2009), Leuven, Belgium*, 2009. (Cited on page 28.)

[120] Wei-Po Lee, Che Kaoli, and Jhih-Yuan Huang. A smart tv system with body-gesture control, tag-based rating and context-aware recommendation. *Knowledge-Based Systems*, 56:167–178, 2014. (Cited on page 28.)

[121] Chia-Chi Wu and Meng-Jung Shih. A context-aware recommender system based on social media. In *International Conference on Computer Science, Data Mining & Mechanical Engineering*, 2015. (Cited on page 28.)

[122] Augusto Q Macedo, Leandro B Marinho, and Rodrygo LT Santos. Context-aware event recommendation in event-based social networks. In *Procs. of the 9th ACM Conf. on Recommender Systems*, pages 123–130. ACM, 2015. (Cited on page 28.)

[123] Choonsung Shin and Woontack Woo. Socially aware tv program recommender for multiple viewers. *IEEE Transactions on Consumer Electronics*, 55(2):927–932, 2009. (Cited on page 28.)

[124] Eli Parser. First monday: What's on tap this month on tv and in movies and books. 2011. (Cited on page 29.)

[125] Sanjay Krishnan, Jay Patel, Michael J. Franklin, and Ken Goldberg. Social influence bias in recommender systems: A methodology for learning, analyzing, and mitigating bias in ratings. RecSys '14, Foster City, CA, USA, 2014. ACM. (Cited on page 29.)

[126] Gui-Ryong Ha and Kyung-Tak Lee. The importance analysis of the selection factors for iptv using ahp. *The Journal of the Korea Contents Association*, 9, 12 2009. (Cited on page 29.)

[127] Daniel Schreiber, Osama Abboud, Sandra Kovacevic, Andreas Höfer, and Thorsten Strufe. Social iptv: a survey on user-acceptance. (Cited on page 29.)

[128] Myung-Joong Kim, Juil Kim, and Sun-Young Park. Understanding iptv churning behaviors: focus on users in south korea. *Asia Pacific Journal of Innovation and Entrepreneurship*, 11:190–213, 08 2017. (Cited on page 29.)

[129] Abdul Nasir and Shahzada Alamgir Khan. The impact of critical iptv factors on customer satisfaction. *Universal Journal of Communications and Network*, 2(4):63 – 69, 2014. (Cited on page 29.)

[130] Point topic iptv statistics - market analysis q2 2013, June 2013. (Cited on page 30.)

[131] Gregor Heinrich. Parameter estimation for text analysis. Technical report, Fraunhofer IGD, Darmstadt, Germnay, 2009. (Cited on pages 39, 42 and 44.)

[132] Jose San Pedro and Alexandros Karatzoglou. Question recommendation for collaborative question answering systems with rankslda. RecSys '14, pages 193–200, New York, USA, 2014. ACM. (Cited on page 40.)

[133] Hongzhi Yin, Yizhou Sun, Bin Cui, Zhiting Hu, and Ling Chen. Lcars: A location-content-aware recommender system. KDD '13, pages 221–229, New York, USA, 2013. ACM. (Cited on pages 40 and 42.)

[134] Rel Guzman Apaza, Elizabeth Vera Cervantes, Laura Cruz Quispe, and José Ochoa Luna. Online courses recommendation based on LDA. In *Proc. the 1st Symposium on Information Management and Big Data - SIMBig 2014, Cusco, Peru, October 8-10, 2014.*, pages 42–48, 2014. (Cited on page 40.)

[135] Tommaso Di Noia, Vito Claudio Ostuni, Jessica Rosati, Paolo Tomeo, and Eugenio Di Sciascio. An analysis of users' propensity toward diversity in recommendations. RecSys '14, pages 285–288, New York, USA, 2014. ACM. (Cited on page 42.)

[136] Christophe Andrieu, Nando de Freitas, Arnaud Doucet, and Michael I. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50(1-2):5–43, 2003. (Cited on page 42.)

[137] Hongzhi Yin, Bin Cui, Ling Chen, Zhiting Hu, and Chengqi Zhang. Modeling location-based user rating profiles for personalized recommendation. *ACM Trans. Knowl. Discov. Data*, 9(3):19:1–19:41, April 2015. (Cited on page 43.)

[138] Zhihui Jin. Lda math gossiping. Technical report, Tencent, China, 2013. (Cited on page 44.)

[139] Mor Naaman, Hila Becker, and Luis Gravano. Hip and trendy: Characterizing emerging trends on twitter. *Journal of the American Society for Information Science and Technology*, 62(5):902–918, 2011. (Cited on page 54.)

[140] Sitaram Asur, Bernardo A Huberman, Gabor Szabo, and Chunyan Wang. Trends in social media: Persistence and decay. *Available at SSRN 1755748*, 2011. (Cited on pages 54 and 85.)

[141] Rong Lu and Qing Yang. Trend analysis of news topics on twitter. *International Journal of Machine Learning and Computing*, 2(3):327, 2012. (Cited on pages 55, 61, 62 and 85.)

[142] Erich Schubert, Michael Weiler, and Hans-Peter Kriegel. Signitrend: scalable detection of emerging topics in textual streams by hashed significance thresholds. In *Proc.s of the 20th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, pages 871–880. ACM, 2014. (Cited on pages 55, 56, 61, 62 and 85.)

[143] J. Lin. Divergence measures based on the shannon entropy. *IEEE Trans. Inf. Theor.*, 37(1):145–151, September 2006. (Cited on page 60.)

[144] Andreas Lommatzsch. Real-time news recommendation using context-aware ensembles. In *Advances in Information Retrieval - 36th European Conference on IR Research, ECIR 2014, Amsterdam, The Netherlands, April 13-16, 2014. Proceedings*, pages 51–62, 2014. (Cited on page 73.)

[145] Benjamin Kille, Andreas Lommatzsch, Roberto Turrin, András Serény, Martha Larson, Torben Brodt, Jonas Seiler, and Frank Hopfgartner. Stream-based recommendations: Online and offline evaluation as a service. In *Proceedings of the Sixth International Conference of the CLEF Association, CLEF'15*, pages 497–517, 2015. (Cited on page 73.)

[146] Patrick Probst and Andreas Lommatzsch. Optimizing a scalable news recommender system. In *Working Notes of CLEF 2016 - Conference and Labs of the Evaluation forum, Évora, Portugal, 5-8 September, 2016.*, pages 669–678, 2016. (Cited on page 73.)

[147] Andreas Lommatzsch, Niels Johannes, Jens Meiners, Lea Helmers, and Jaschar Domann. Recommender ensembles for news articles based on most-popular strategies. In *Working Notes of CLEF 2016 - Conference and Labs of the Evaluation forum, Évora, Portugal, 5-8 September, 2016.*, pages 657–668, 2016. (Cited on page 73.)

[148] Andreas Lommatzsch. Real-time recommendations for user-item streams. In *Proc. of the 30th Symposium On Applied Computing, SAC 2015*, SAC '15, pages 1039–1046, New York, NY, USA, 2015. ACM. (Cited on page 74.)

[149] Alan Said, Alejandro Bellogín, Jimmy Lin, and Arjen P. de Vries. Do recommendations matter?: news recommendation in real life. In *Computer Supported Cooperative Work, CSCW '14, Baltimore, MD, USA, February 15-19, 2014, Companion Volume*, pages 237–240, 2014. (Cited on pages 75 and 76.)

[150] Tomás Kliegr and Jaroslav Kuchar. Benchmark of rule-based classifiers in the news recommendation task. In Josiane Mothe, Jacques Savoy, Jaap Kamps, Karen Pinel-Sauvagnat, Gareth J. F. Jones, Eric SanJuan, Linda Cappellato, and Nicola Ferro, editors, *CLEF*, volume 9283 of *Lecture Notes in Computer Science*, pages 130–141. Springer, 2015. (Cited on page 76.)

[151] Jaroslav Kuchar and Tomás Kliegr. InBeat: Recommender System as a Service. In *Working Notes for CLEF 2014 Conference, Sheffield, UK, September 15-18, 2014*, pages 837–844, 2014. (Cited on page 76.)

[152] Gebrekirstos Gebremeskel and Arjen P. de Vries. The degree of randomness in a live recommender systems evaluation. In *Working Notes of CLEF 2015 - Conference and Labs of the Evaluation forum, Toulouse, France, September 8-11, 2015.* CEUR, 2015. (Cited on page 76.)

[153] Doychin Doychev, Rachael Rafter, Aonghus Lawlor, and Barry Smyth. News recommenders: Real-time, real-life experiences. In *Proceedings of UMAP 2015*, pages 337–342, 2015. (Cited on page 76.)

[154] Xuning Tang and Christopher C. Yang. Tut: A statistical model for detecting trends, topics and user interests in social media. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 972–981, New York, NY, USA, 2012. ACM. (Cited on page 85.)

[155] Bruno Cardoso and João Magalhães. Google, bing and a new perspective on ranking similarity. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 1933–1936, New York, NY, USA, 2011. ACM. (Cited on page 87.)

[156] William Webber, Alistair Moffat, and Justin Zobel. A similarity measure for indefinite rankings. *ACM Trans. Inf. Syst.*, 28(4):20:1–20:38, November 2010. (Cited on pages 87 and 88.)

[157] Ronald Fagin, Ravi Kumar, and D. Sivakumar. Comparing top k lists. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '03, pages 28–36, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics. (Cited on page 88.)

[158] Pavel B. Brazdil and Carlos Soares. *A Comparison of Ranking Methods for Classification Algorithm Selection*, pages 63–75. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000. (Cited on page 97.)

[159] S.M. Abdulrahman, P. Brazdil, J.N. van Rijn, and J. Vanschoren. Speeding up algorithm selection using average ranking and active testing by introducing runtime. *Machine Learning*, 107:79–108, 1 2018. (Cited on page 97.)

[160] Tri Doan and Jugal Kalita. *Algorithm Selection Using Performance and Run Time Behavior*, pages 3–13. Springer International Publishing, Cham, 2016. (Cited on page 98.)

[161] Bao Lin Lin, Xiaoyan Sun, and Sana Salous. Solving travelling salesman problem with an improved hybrid genetic algorithm. *Journal of computer and communications.*, 4(15):98–106, November 2016. (Cited on page 99.)

[162] Salisu Abdulrahman and Pavel Brazdil. Measures for combining accuracy and time for meta-learning. *CEUR Workshop Proceedings*, 1201:49–50, 01 2014. (Cited on page 112.)

[163] Salisu Mamman Abdulrahman, Brazdil Pavel, Zainon Wan Mohd Nazmee Wan, and Adamu Alhassan. Simplifying the algorithm selection using reduction of ranking of classification algorithms. In *Proceedings of the 2019 8th International Conference on Software and Computer Application*, pages 140–148. ACM, 2019. (Cited on page 112.)