STUDIES IN COMPUTER SCIENCE NO 9, DOCTORAL DISSERTATION

DIMITRIS PARASCHAKIS

SOCIOTECHNICAL ASPECTS OF AUTOMATED RECOMMENDATIONS

Algorithms, Ethics, and Evaluation

MALMÖ UNIVERSITY

SOCIOTECHNICAL ASPECTS OF AUTOMATED RECOMMENDATIONS: ALGORITHMS, ETHICS, AND EVALUATION

Malmö University
Studies In Computer Science No 9,
Doctoral dissertation

ISBN 978-91-7877-074-8 (print)
ISBN 978-91-7877-075-5 (pdf)
DOI 10.24834/isbn.9789178770755
Holmbergs, Malmö 2020

DIMITRIS PARASCHAKIS

SOCIOTECHNICAL ASPECTS OF AUTOMATED RECOMMENDATIONS

Algorithms, Ethics, and Evaluation

Department of Computer Science
Faculty of Technology and Society
Malmö University, 2020

Studies in Computer Science

Faculty of Technology and Society
Malmö University

1. Jevinger, Åse. Toward Intelligent Goods: Characteristics, Architectures and Applications, 2014, Doctoral dissertation.
2. Dahlskog, Steve. Patterns and Procedural Content Generation in Digital Games: Automatic Level Generation for Digital Games Using Game Design Patterns, 2016, Doctoral dissertation.
3. Fabijan, Aleksander. Developing the Right Features: the Role and Impact of Customer and Product Data in Software Product Development, 2016, Licentiate thesis.
4. Paraschakis, Dimitris. Algorithmic and Ethical Aspects of Recommender Systems in E-commerce, 2018, Licentiate thesis.
5. Hajinasab, Banafsheh. A Dynamic Approach to Multi Agent Based Simulation in Urban Transportation Planning, 2018, Doctoral dissertation.
6. Fabijan, Aleksander. Data-driven Software Development at Large Scale, 2018, Doctoral dissertation.
7. Bugeja, Joseph. Smart Connected Homes: Concepts, Risks, and Challenges, 2018, Licentiate thesis.
8. Alkhabbas, Fahed. Towards Emergent Configurations in the Internet of Things, 2018, Licentiate thesis.
9. Paraschakis, Dimitris. Sociotechnical Aspects of Automated Recommendations: Algorithms, Ethics, and Evaluation, 2020, Doctoral dissertation.

Electronically available at:
http://mau.diva-portal.org

Dedicated to my family

ABSTRACT

Recommender systems are algorithmic tools that assist users in discovering relevant items from a wide range of available options. Along with the apparent user value in mitigating the choice overload, they have an important business value in boosting sales and customer retention. Last, but not least, they have brought a substantial research value to the algorithm developments of the past two decades, mainly in the academic community. This thesis aims to address some of the aspects that are important to consider when recommender systems pave their way towards real-life applications.

We begin our investigation by assessing the adoptability of popular recommendation algorithms by e-commerce platforms, and perform the comparative evaluation of these algorithms on real sales data provided by Apptus Technologies. Based on the conducted survey and offline experiments, our research clarifies which algorithms are particularly useful for sales data.

The realistic modeling and evaluation of recommender systems is another issue of utmost importance. Over the years, the field has been gradually moving away from the oversimplified matrix completion abstraction to more pragmatic modeling paradigms, such as *sequential*, *streaming*, and *session-aware/session-based* recommender systems (or all at once). Despite the rapidly increasing body of work in each of those directions, there is a need for more unified algorithmic solutions and evaluation frameworks supporting them. To this end, we propose two recommender systems for streaming session data, as well as a new benchmarking/prototyping tool based on the streaming framework `Scikit-Multiflow`.

Finally, a somewhat overlooked aspect of recommender systems is their ethical implications. When a recommender is intended to leave the lab and be deployed to real users, the purely accuracy-oriented algorithmic approach is no longer sufficient. A deployed recommender system must also guarantee its compliance with the societal and legal norms, such as anti-discrimination laws, GDPR, privacy, fairness,

etc. To aid the development of ethics-aware recommender systems, we provide a holistic view on potential ethical issues that may arise at various stages of the development process, and advocate the provision of user-adjustable ethical filters. Among all ethical matters, algorithmic fairness stands on its own as a rapidly developing sub-field of machine learning, which has recently made its entry to the realm of recommender systems. We contribute to this research direction by formulating and solving the problem of preferentially fair matchmaking in speed dating with minimal accuracy compromises.

PUBLICATIONS

Included publications

Paper I D. Paraschakis, B. J. Nilsson, and J. Holländer,
"Comparative Evaluation of Top-N Recommenders in e-Commerce: an Industrial Perspective" [221]
In proceedings of the 14th IEEE International Conference on Machine Learning and Applications (ICMLA), 2015

Paper II D. Paraschakis,
"Towards an Ethical Recommendation Framework" [217][1]
In proceedings of the 11th IEEE International Conference on Research Challenges in Information Science (RCIS), 2017

Paper III B. Brodén, M. Hammar, B. J. Nilsson, and D. Paraschakis,
"A Bandit-Based Ensemble Framework for Exploration / Exploitation of Diverse Recommendation Components: An Experimental Study within e-Commerce" [32]
In ACM Trans. Interact. Intell. Syst. 10, 1, Article 4, Special Issue "Highlights of IUI 2018", 2019

Paper IV D. Paraschakis and B. J. Nilsson,
"`FlowRec`: Prototyping Session-based Recommender Systems in Streaming Mode" [218]
In proceedings of the 24th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), 2020

Paper V D. Paraschakis and B. J. Nilsson,
"Matchmaking Under Fairness Constraints: a Speed Dating Case Study" [219]
In proceedings of the International Workshop on Algorithmic Bias in Search and Recommendation (held as part of ECIR'20), 2020

---

[1] Best paper award

Related publications that are not included in the thesis

- D. Paraschakis,
  "Recommender Systems from an Industrial and Ethical Perspective" [216]
  In proceedings of the 10th ACM International Conference on Recommender Systems (RecSys), 2016

- B. Brodén, M. Hammar, B. J. Nilsson, and D. Paraschakis,
  "Bandit Algorithms for e-Commerce Recommender Systems: Extended Abstract" [29]
  In proceedings of the 11th ACM International Conference on Recommender Systems (RecSys), 2017

- B. Brodén, M. Hammar, B. J. Nilsson, and D. Paraschakis,
  "An Ensemble Recommender System for e-Commerce" [30]
  In proceedings of the 26th Benelux Conference on Machine Learning (Benelearn), 2017

- B. Brodén, M. Hammar, B. J. Nilsson, and D. Paraschakis,
  "Ensemble Recommendations via Thompson Sampling: an Experimental Study within e-Commerce" [31]
  In proceedings of the 23rd ACM International Conference on Intelligent User Interfaces (IUI), 2018

- D. Paraschakis and B. J. Nilsson,
  "On Preferential Fairness of Matchmaking: a Speed Dating Case Study" [220]
  In proceedings of the 18th International Conference on the Ethical and Social Impacts of ICT (ETHICOMP), 2020

Personal contribution

In all the included papers, the author of the thesis was the main contributor with respect to research planning, execution, and reporting.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my supervisor Bengt J. Nilsson for all the effort and patience he has put in mentoring my studies and guiding my research. I thoroughly enjoyed our brainstorming sessions, often necessary to knock me out of my "local optima".

Special thanks to Jan Persson for being my examiner and making my PhD journey possible. I am also grateful to all my advisors for their valuable input at various stages of my research: Yuanji Cheng, Mikael Hammar, Helena Holmström Olsson, Paul Davidsson.

A crucial part of my research was conducted in collaboration with Apptus Technologies. The time spent in their research team with Björn Brodén and Mikael Hammar has improved my technical skills and understandings of industry-level recommendation engines. I truly appreciate it.

I am deeply thankful to Johan Holmgren, as well as to Nancy Russo, Annabella Loconsole, Edward Blurock, and all others who in some way contributed to the pedagogical aspect of my studies.

I would also like to mention Christina Bjerkén, Per Jöhnsson and Åse Jevinger for their great job as directors of doctoral studies.

Finally, I would like to thank my family for their support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

PART I

COMPREHENSIVE SUMMARY

1 INTRODUCTION

1.1 Preamble

> *The Web, they say, is leaving the era of search and entering one of discovery. What's the difference? Search is what you do when you're looking for something. Discovery is when something wonderful that you didn't know existed, or didn't know how to ask for, finds you.*
> - CNN Money

Recommender systems (RS) have become indispensable and ubiquitous tools for filtering and surfacing the relevant information in the digital world. They address the important problem of *choice overload*, which has proven detrimental to our emotional and psychological well-being [262]. More specifically, recommender systems help their users to [189]:

- *decide*, by predicting a relevance score (e.g. a rating) for an item
- *explore*, by suggesting similar items for a given item
- *compare*, by personalizing the ranking of a given list of items for a user
- *discover*, by finding unknown but relevant items for a user

Helping users to satisfy their information needs in an effective and efficient way is a highly rewarding task for the businesses as well: reportedly, 35% of sales on Amazon and 75% of downloads on Netflix result from recommendations [9, 179]. Apart from e-commerce, the application of recommender system spans a host of other domains, including multimedia (movies, music), food (restaurants, recipes), social networks, mobile apps, jobs, and many more. The experimental work presented in this thesis covers the e-commerce, dating, news, and travel domains.

With this thesis, we aim to contribute to some of the important practical challenges of present-day recommender systems, namely the issues of ethics, algorithms, and evaluation. Our contributions range from theoretical explorations to open-source implementations. While not immediately apparent, the algorithmic and the ethical aspects of recommender systems are often interrelated. For example, a recommendation algorithm that relies on user profiling may have ethical implications, e.g., unwanted tracking. On the other hand, the prohibition of user profiling for ethical reasons would require a different algorithmic approach — perhaps less personalized as in anonymous session-based recommendations. Another example of the interrelation between the algorithmic and ethical aspects of RS is the connection between multi-stakeholder recommender systems and multi-sided fairness [1]. Therefore, it is important to understand the practical challenges associated with both the algorithmic and the ethical counterparts, and make a connection where possible. This is attempted in the rest of the section, serving as a motivation for our research questions.

1.2 From academia to industry

> *Academia solves simple problems with complex methods.*
> *Industry solves complex problems with simple methods.*
> - Björn Brodén, Apptus Technologies

Despite that recommender systems have been known since early 1990s, the academic interest toward this field has dramatically increased after the announcement of the Netflix Prize[2] contest in 2006. With a million dollars prize money, Netflix went on a quest for recommendation algorithms capable of surpassing the accuracy of Netflix's own algorithm by 10%. This contest spurred active research within the field, yielding a variety of algorithmic offerings over the past 15 years. Surprisingly, the prize-winning algorithm was never put to real use: Netflix concluded that the measured accuracy gains "did not seem to justify the engineering effort needed to bring them into a production environment" [9]. This interesting fact calls for a deeper analysis of the gap between research contributions and commercial applications, which until recently has not been a frequent discussion topic in the RS literature.

[2] `http://www.netflixprize.com/`

The Netflix problem was essentially formulated as a *rating prediction* task. This has the following rationale: if we can accurately predict user's ratings for unseen movies, then those candidates that receive the highest predicted rating can be recommended to the user. Due to this problem formulation, and because of the early availability of datasets containing movie ratings (e.g. MovieLens, EachMovie), the vast number of early recommender systems have been modelled as rating predictors and evaluated as such. Conversely, many commercial systems only provide the so-called "top-N" recommendations constructed from implicit interactions such as clicks, purchases, and so on. This paradigm typically requires different approaches to algorithms and evaluation [52]. We focus on this type of recommendations in this thesis.

Considering the general lack of publicly available e-commerce datasets, researchers have acknowledged the need for case studies on real-life sales data to better understand the specifics of these datasets and the factors that are important for deploying recommender systems in retail [230]. Our research starts with such a case study, accompanied with a survey of e-commerce platforms featuring recommendation engines (Paper I).

1.3 From matrix completion to session modeling

As long as the Matrix exists, the human race will never be free.
- Morpheus, The Matrix movie

Paraphrasing the above quote, "as long as matrix completion exists, the RecSys community will never be free from limitations". *Matrix completion* has long remained the primary abstraction of the recommendation problem in the academic field. This was the case with the Netflix Prize, too. The idea is to have a matrix of user-item ratings (or other types of interactions), where the computational task is to predict the missing entries of the matrix. Some of the known entries are held out (often randomly), in order to enable the evaluation using error metrics (e.g. RMSE, MAE) or accuracy metrics (e.g. precision, recall). This is an attractive setup for academic research as it offers standardized evaluation, reproducibility, and mathematical convenience [52, 122], particularly with matrix factorization methods [124, 154, 242, 250]. Although dominant in the literature, this abstraction is admittedly oversimplified as it relies on a single type of interaction and disregards the sequential patterns of interactions. Evaluation-wise, it has been acknowledged that the withheld ratings are not representative of the actually missing ratings, which may mislead the performance assessment of a system [122].

In practice, user activity logs are organized into time-ordered sessions. Recognizing the needs for the direct modeling of session data in RS, the research community has recently witnessed a notable paradigm shift to *session-aware* and *session-based* recommendations. The former paradigm models long-term user preferences across sessions, whereas the latter one predicts the short-term intention of a user within a session [303]. The session-based approach is practical as the majority of users in real-world scenarios remain anonymous due to being either firstcomers, or logged out visitors (sometimes deliberately so as to avoid tracking) [119]. A non-reliance on user profiles in session-based recommendations therefore makes user privacy less of a problem.

The increasing availability of sessionized datasets [21, 33, 137, 143, 288] during the past few years has significantly spurred the research in this direction. Different from the matrix completion approach, the key task of session-based recommendations is to predict the next likely event(s) given the sequence of previous events in a session. Another difference is that a single item can appear in different types of events, e.g. the same item is first clicked, then added to a shopping cart, and then bought. Event-awareness enables the identification of complementary and substitute items, which are important for personalizing recommendations in e-commerce. For example, it is customary to view frequent co-purchases as signals of complementary items [316]. Another possibility that session modeling offers is reminding users of previously visited items. Such recommendations have shown significant business value [119]. For the reasons above, session modeling better resembles the actual human-recommender interaction as compared to the matrix completion setup. It also necessitates a different algorithmic approach to recommendation and evaluation. Due to the sequential nature of the problem, solutions based on Recurrent Neural Networks (RNN) have become especially attractive for this task [47, 96, 108, 232, 234, 280]. Very good results have also been demonstrated with conceptually and computationally simpler methods, such as kNN and sequential rules [120, 173]. In this thesis, we explore alternative solutions to session-based recommendations that are based on multi-arm bandits [32] and decision trees [218].

Session-based recommendations are naturally evaluated using sequential protocols [232], where the number of subsequent events to predict is dictated by the application needs. One of the most common scenarios is *next-item* prediction, which is meaningful in several domains such as news, music, and advertising. We employ this protocol in Paper IV. In *look-ahead* prediction, the recommendations are evaluated against the sequence of next n events in the session. In certain scenarios, it is useful to predict *all* ground truth events regardless of their sequence, which is typical of matrix completion setups. In e-commerce applications, for instance, it is often desirable to predict which items will eventually be bought in the current session [21]. This scenario is explored in Paper III. Recommending all ground truth items is also done in Paper V, albeit in a supervised learning setting and a different application domain (speed dating).

1.4 From batch to streaming

> *Amid the tech wiring, data streams and programmatic connections dancing around us, simple ideas matter more than ever.*
> - Pete Blackshaw, Nestlé

In industrial contexts, offline evaluation is often used to determine whether an algorithm is to be further "A/B tested" on real users. Consequently, offline evaluation protocols should strive to approximate the online settings as closely as possible. The RS evaluation in academic research has traditionally been done in *batch* mode, i.e. using static train-test splits of pre-recorded data [94]. This approach assumes that all data are available at once. Presently, even the aforementioned state-of-the-art RNN-based recommender systems, which have widely adopted sequential evaluation protocols, rely on batch training on static chunks of historical data before entering the testing phase. This limits their applicability in industrial contexts, where one of the requirements is the ability to make useful recommendations at the time of the system's launch, when insufficient amounts of data have been collected yet [189]. Upon the arrival of new data, a recommender system would have to periodically re-learn the entire model, which can be computationally impractical in industrial applications.

In the real world, recommendations are provided sequentially in response to the events arriving from a *stream* of data. *Stream-based* (or *streaming*) recommender systems adopt the online learning[3] paradigm, according to which the learner is tested and trained incrementally (precisely in this order) as soon as new data become available. This is efficient because each data point is typically accessed only once and then discarded, whereas the existing model is merely adjusted to stay up-to-date.

Because the importance of old data gradually diminishes with respect to new data, this approach is naturally suited for the *concept drift* adaptation, which is relevant to RS in a number of ways. First and foremost, it addresses the volatility of user preferences, which are subject to change over time [184]. Secondly, it can deal with highly dynamic domains (news, advertising, videos, etc.), in which new items constantly emerge while previous items rapidly lose their relevance [138, 166]. In Paper III we deal with bandit ensembles [32, 281], for which the component recommenders may also change their behavior as they process more and more data points from the stream, which may impact the overall performance of the ensemble. In this case, relative weights or other parameters of component learners can be dynamically adjusted to accommodate the drift [296].

Because of their practical value, the importance of streaming recommendations has been increasingly recognized in recent years [275]. A number of incremental RS have been proposed from various algo-

[3]not to be confused with online evaluation, e.g. A/B testing

rithmic families, such as neighborhood-based methods [205, 275], ensemble learning methods [298, 299], matrix factorization [12, 175, 257, 279, 297], tensor factorization [269, 322], multi-arm bandits [39, 158, 164, 166], and some combinations thereof [136, 302, 326]. However, the majority of existing stream-based RS have been designed for the traditional recommendation problem (i.e. the matrix completion abstraction), rather than for session-based recommendations [96]. The present thesis contributes to this emerging research direction.

One of the benefits of stream-based learning in RS is that it allows for real-time performance monitoring of the algorithm's evolution over time, across a variety of evaluation metrics [296]. This enables the experimenter to get a clearer picture of how algorithms perform relative to each other on various segments of the dataset (e.g. during cold-start), as well as to understand the peculiarities of the dataset itself (e.g. signs of seasonal effects). Presently, very few libraries/benchmarking frameworks for streaming RS are publicly available for academic research. Some of them are limited to the matrix completion problem [83, 140], whereas others have been developed for a particular application domain [126, 264]. A general-purpose open source machine learning library for data streams has recently been released under the name `Scikit-Multiflow`[4]. It offers a rich set of tools (incremental algorithms, evaluation protocols, change detectors) to facilitate the research on stream learning. According to its authors [198], `Scikit-Multiflow` is intended to complement the popular machine learning library `Scikit-learn` [226], and become a Python-based equivalent to other popular stream learning libraries, namely `MOA` [26] and `MEKA` [238]. Although not directly applicable to the recommendation problem out of the box, `Scikit-Multiflow` opens new possibilities for the future developments of stream-based recommender systems in a more standardized and replicable way. We explore this potential in Paper IV.

[4] **https://scikit-multiflow.github.io/**

1.5 From algorithms to ethics

There is no right way to do a wrong thing.
- Harold S. Kushner, writer

Whenever algorithms get involved in decision making, the issues of ethics inevitably come to the forefront. Recommender systems are no exception. There are concerns that today's research practice in personalization and recommender systems is dangerously unbalanced, as it often puts commercial success above considerations of the moral impact of this technology [147]. Yet, the discussion around recommendation ethics as such remains very sparse and fragmented. Earlier works that directly address ethics-aware recommendations [104, 245, 270, 282] focus on some specific moral issues arising in specific recommendation scenarios. Prior to the ethical framework presented in this thesis (Paper II), the holistic view on the ethics of RS was lacking in the field. Only recently, Milano et al. presented a survey of the ethical challenges in RS [191], and extended it to the multi-stakeholder environments [190]. Knowing what ethical issues may appear at each stage of the RS development cycle would facilitate the emergence of more user-friendly, privacy-preserving, non-discriminatory, and fair recommender systems. To this end, our investigation of ethics-related incidents scarcely reported in the RS literature has been necessary to draw a roadmap for the potential ethical issues and their possible solutions.

One such motivating example brings us back to the Netflix Prize contest. Two years after its public release, the Netflix dataset was deanonymized via a linking attack [203], putting a privacy of its 500,000 uses at risk. This resulted in a lawsuit and put an end to the planned Netflix Prize sequel due to the raised privacy concerns [147]. Clearly, personal privacy is one of the biggest issues in the ethical discourse around the practices of data analytics, having to do with the leakage of confidential personal information encoded in user profiles. As mentioned earlier, session-based recommendations alleviate this problem by respecting the anonymity of a user. The privacy implications of RS have been thoroughly studied by Friedman et al. [82]. To derive a classification of recommendation ethics, the research has to be extended to other moral issues beyond privacy and anonymization in the context of RS.

The three biggest challenges of today's Web [75], outlined in a letter penned by the Web's creator Tim Berners-Lee, are (1) the loss of control over personal data, (2) the easy spread of misinformation, and (3) the lack of transparency. Apparently, all these are ethics-related issues, since they pose a threat to the general moral principle of "do no harm". It is not hard to find examples in the RS literature that relate to each of the above. For instance, the Netflix case mentioned earlier exemplifies the "failure of anonymization" [213], which is responsible for issue (1). Issue (2) has become particularly relevant in the light of the controversies around the U.S. presidential election of 2016 and the Cambridge Analytica scandal of 2018, which makes a strong case about the vulnerability of news recommenders and news feed algorithms in disseminating fake news [196]. Finally, the lack of transparency stated in issue (3) makes it possible to hide the evidence of algorithmic discrimination, leading to such known incidents as recommendations of lower paying jobs to female candidates or higher priced flights to MacBook owners [243].

The FATML[5] community, standing for Fairness, Accountability, and Transparency in Machine Learning, has formulated a set of five principles for determining the social and moral impact of algorithmic decision making: responsibility, explainability, accuracy, auditability, and fairness. Their purpose is to "help developers design and implement algorithmic systems in publicly accountable ways" [62]. We briefly overview these principles in Section 2.7.4. With the research presented in this thesis, we hope to draw attention to these topics in the narrower field of RS, which clearly represents a class of such algorithmic systems particularly susceptible to undesirable social and moral effects.

Many ethical issues in machine learning can be addressed by either technological or normative means (or both). Normative solutions are implemented via regulatory incentives, such as the General Data Protection Regulation (GDPR)[6] in Europe, or the California Consumer

[5] **https://www.fatml.org/**
[6] **https://gdpr-info.eu/**

Privacy Act (CCPA)[7] in the U.S. As these solutions fall on the legal side of ethics, we leave them outside the scope of this thesis.

Some of the important moral questions of algorithmic decision making can also be approached algorithmically. For example, the questions of user privacy and fairness in RS are addressed with privacy-preserving collaborative filtering [161, 229, 320] and fairness-aware recommender systems [36, 132, 169, 319, 327, 328], respectively. Many notions of fairness from the social sciences literature have been translated to mathematical form, making them attractive for formulating computational problems. In practical applications, achieving fairness inevitably implies resolving the accuracy-fairness tradeoff [168, 310]. For instance, a recommender system for a dating app must be able not only to predict highly probable matchings, but also to comply with fair candidate selection policies. We investigate this direction in Paper V.

In our view, recommendation ethics would benefit from interdisciplinary approaches, where existing legal policies are backed with special-purpose technology designed to protect the moral rights and liberties of RS users. At this stage, there seems to be a lack of consensus amongst corporate and academic researchers regarding the "solutions tookit" with exact methods for enabling algorithmic accountability [75]. In this thesis, we propose an "ethical toolbox" for RS users, giving them the necessary instruments to control the morally sensitive aspects of a recommendation engine. The provision of such tools aims to complement the existing technical means discussed above, thus promoting the idea of ethics-awareness by design. The results of the feasibility study reported in Paper II show promising prospects for further research and development of such user-centric toolkits.

1.6 Research questions

Each of the included papers of this thesis poses a distinct research question in relation to the algorithmic or the ethical aspect of recommender systems (or both). For each of them, we formulate a general research question (RQ), accompanied with more specific sub-RQs. For

[7] `https://oag.ca.gov/privacy/ccpa`

convenience, research questions are numbered in the same order as the corresponding papers that address them.

RQ I How to assess the receptiveness of the e-commerce domain to the algorithmic innovations in the field of recommender systems?

- **RQ I (a)** Which recommendation algorithms perform well on sales data?
- **RQ I (b)** Which recommendation algorithms are favored by industrial e-commerce platforms?

RQ II What are the ethical challenges that impact the design and use of recommender systems, and what are the possible solutions?

- **RQ II (a)** How to aid morality in recommender systems through user engagement?

RQ III How to build a session-based context-aware bandit ensemble of elementary recommendation components with non-stationary rewards?

- **RQ III (a)** How does Thompson Sampling compare to other bandit policies for orchestrating the ensemble?
- **RQ III (b)** How to prime the sampler with prior knowledge, and what effect does it have?
- **RQ III (c)** How to personalize the ensemble with anonymous users, and what effect does it have?

RQ IV How to adapt the `Scikit-Multiflow` streaming framework for rapid prototyping of session-based recommender systems?

- **RQ IV (a)** How to utilize the underlying stream learners for the recommendation task?

RQ V How to define and reason about the fairness of matchmaking in the context of speed dating, with respect to the expressed preferences for sensitive attributes of users?

- **RQ V (a)** How to measure *preferential fairness*?
- **RQ V (b)** How to resolve the fairness-accuracy trade-off?

Outline The remainder of the thesis' comprehensive summary is organized as follows. In the next section, we provide the necessary theoretical background and build the intuition for understanding the rest of the thesis. After that, we describe our methodological approach with a detailed overview of each research method involved. We then go on with the discussion of our contributions to each of the research questions formulated above. The results of the thesis and the general conclusions drawn from them are summarized in the final section.

The second part of the thesis contains the compilation of the original research papers.

2 BACKGROUND

This section provides the relevant background essentials of recommender systems that set the stage for the remainder of the thesis.

2.1 Recommendation problem

Recommender systems traditionally pursue two alternative fundamental objectives:

Objective 1: estimate the utility function that predicts *how* a user will *like* an item (i.e. rating prediction)

Objective 2: estimate the utility function that predicts *whether* the user will *choose* an item (i.e. item prediction)

The recommendation problem itself is usually defined as either *best item* or *top-N recommendation* [59]. The first task aims to suggest the most interesting new item for a target user, whereas the second task aims to suggest top-N such items. A recommendation is considered successful when the suggested item is consumed by the user, where the definition of consumption is application-dependent (e.g. buying an item, watching a movie, listening to a song, etc.).

2.2 Types of feedback

Recommender systems rely on two types of user feedback: *explicit* and *implicit*.

2.2.1 Explicit feedback

This type of feedback is provided directly in the form of *ratings* on a numeric or ordinal scale, e.g. when a user rates a watched movie. Another example is the binary "like/dislike" indication of preference. Explicit preference acquisition makes this feedback unambiguous and reliable. However, there are a few drawbacks:

- Rating data are typically sparse, as only a small fraction of consumed items are rated.

- Rating functionality is not always supported by commercial platforms.

- Rating-based recommendations are given under the assumption that users are only interested in top-rated items. However, this does not always hold in practice. For example, cheaper items tend to receive lower ratings than the expensive ones, but might nevertheless be preferred more often due to being more affordable.

2.2.2 Implicit feedback

This type of feedback is collected implicitly from various user interactions on a website, such as clicks, purchases, search queries, etc. [153], which act as a proxy for the user's preference. The feedback that the system receives when such an event is registered as a result of successful recommendation takes the form of *rewards*. In most cases, this implies binary preference, e.g. $reward = 1$ if the item is bought, $reward = 0$ if the item is not bought (a.k.a. *positive-only* feedback). The main advantage of this type of feedback is that it is generally denser and easier to collect than the explicit one, since no direct user input is required. However, this does not apply to all types of data. For instance, purchase events are even sparser than rating data, since a user cannot rate an item until he/she buys it. Furthermore, implicit feedback is more ambiguous, since $reward = 1$ does not convey how much the user liked the bought item, and $reward = 0$ does not distinguish between items that were not examined, and items that were examined but not preferred. Therefore, it is generally less reliable than explicit feedback, and should be treated more carefully.

2.3 Problem abstractions

The recommendation problem is typically reduced to a suitable abstraction depending on the input data and the task at hand. We outline some of the popular and relevant problem abstractions encountered in the RS literature.

2.3.1 Matrix completion

Matrix completion [237] is the most traditional and widely used abstraction for modeling recommender systems. Given a set of m users $U = \{u_1, \ldots, u_m\}$ and a set of n items $I = \{i_1, \ldots, i_n\}$, they can be represented as dimensions of a $m \times n$ matrix, whose cells contain user-item interactions (either observed or missing). These can be ratings or positive-only observations such as purchases. The user-item matrix is typically incomplete and sparse [303]. The goal is to accurately estimate the missing entries of the matrix. The vast majority of these methods model user-item associations based on their low-dimensional representations in a latent space.

2.3.2 Learning to rank

Ranking approaches directly learn the ordering of preferences over recommendable items [134]. This is well-motivated by the fact that the success of a recommender system essentially lies in getting the top-N items right. The methodology comes in three flavors: pointwise, pairwise, and listwise ranking. Matrix completion can be seen as a pointwise learning to rank method, which minimizes the ranking loss function defined on an individual relevance judgement, i.e. $f(u, i) \to \mathbb{R}$. It can be achieved via classification or regression. With pairwise ranking, i.e. $f(u, i_1, i_2) \to \mathbb{R}$, the loss function is defined on pairwise preferences, where the goal is to minimize the number of inversions. This can be solved with pairwise classification. Listwise ranking, i.e. $f(u, i_1, \ldots, i_n) \to \mathbb{R}$, directly optimizes some top-weighted ranking measure such as MAP, NDCG, or MRR (defined in Section 2.6.2). This can be achieved via genetic programming, boosting, simulated annealing, etc.

2.3.3 Multi-arm bandit

A less mainstream, but an increasingly popular abstraction is that of a *multi-arm bandit* (MAB) game: given a set of actions (or bandit "arms") each having a fixed but unknown probability of reward, the player at each time step selects an arm to pull with the aim to maximize his/her cumulative reward. An arm pulling strategy hence needs to balance between exploration and exploitation of available arms. Translated to the RS scenario, the task is to estimate the relevance of less known items (exploration), while ensuring that relevant items continue to get recommended (exploitation) [172]. The selection of an item to recommend is done according to some MAB policy. Alternatively, bandit arms can represent distinct recommendation models [77, 281]. The exploration-exploitation dilemma fits the sequential nature of recommendations, and is very appropriate in dynamic domains where the set of users and their preferences change over time [166].

The abstraction can be extended to the *contextual* multi-arm bandit problem [45, 164, 281], where the choice of an action at a given time step also depends on the observed contextual information associated with a user-item pair. This is intuitively sensible, since the perceptions of different users on the same item can vary significantly [164]. Huang & Lin [114] study contextual bandits with delayed reward attribution, which is a typical scenario in many practical applications.

2.4 Recommendation paradigms

We now review some of the popular recommendation paradigms that have emerged during the past decade. They share certain commonalities, especially when it comes to the evaluation methodologies.

2.4.1 Context-aware/time-aware recommendation

Traditional algorithmic approaches consider two entities for generating recommendations: users and items. *Context-aware* recommender systems go further by incorporating various contextual factors into the recommendation process, so that recommendations can be provided in specific circumstances [2]. For example, the choice of a movie to watch may depend on where, when, and with whom it will happen. These factors allow for a highly personalized user experience. Formally, context-awareness employs a scoring function $f(u, i, c) \to \mathbb{R}$, where c denotes some contextual information associated with the application. The added dimension(s) results in a so-called *multiverse* recommendation abstraction, which extends to tensor completion [133]. In general, context-awareness can be introduced in three ways: pre-filtering (contextualizing the input), post-filtering (contextualizing the output), or in-filtering (contextualizing the RS function) [2].

Time information is one of the most useful and easy to collect types of contexts, which may lead to substantial improvements in recommendation accuracy [40]. For this reason, *time-aware* RS have gained significant traction in the RecSys community. By keeping track of each event's time of occurrence, a recommender system can establish the right time to recommend a particular product [303]. Time awareness also enables sequence modeling, and various data ageing and forgetting mechanisms. Time can be represented as either a continuous variable holding timestamps of events, or a categorical variable encoding various periods of interest, e.g. workday/weekend.

2.4.2 Session-aware/session-based recommendation

Different from the matrix completion setup, *session-aware* recommender systems aim to capture the user's intention within and across their browsing sessions [303]. A *session* starts when the user enters the website and ends after an extensive period of user inactivity (the widely adopted standard is 30 minutes [173, 303]). It contains a temporally ordered sequence of user-item interactions.

Session-based recommender systems [119] are limited to the scope of an active session, thus adapting to the user's short-term intent. This is the most common type of session awareness in RS due to the fact that the majority of online users remain unidentified [289, 308]. The typical goal of session-aware/session-based RS is to predict the user's immediate next action(s) in the current session based on their long-term and/or short-term behavior.

2.4.3 Sequence-aware recommendation

The focus of *sequence-aware* recommender systems is on the sequential order of user-item interactions [232]. The order of recommendations also plays a crucial role in many situations. For instance, recommending a battery after purchasing a camera makes perfect sense, but it would not work as well in reverse. Sequence-aware recommendations can be employed to solve a range of tasks, e.g. presenting complementary/alternative items and making reminders of replenishing consumables in e-commerce; suggesting playlist continuations in music services; detecting consumption trends on an individual as well as a community level. This paradigm is different from the traditional approaches that establish the relevance of a candidate item without considering the previously consumed items of a user. Therefore, sequence-aware RS are most useful for interactional context adaptation in session data, where the sequence of user actions can help to infer his/her current intent and detect potential interest drifts. In this respect, it comes hand in hand with the previous paradigm, but in principle can also be used for matrix completion setups (e.g. [317, 323]). Sequence-aware RS work by capturing primarily sequential, but also co-occurrence patterns in the user's browsing history. Being designed for temporal sequence modeling, Recurrent Neural Networks (RNN) [108, 234, 280] have become a preferred approach for sequence-aware RS.

2.4.4 Stream-based recommendation

In practical applications, recommender systems are challenged with the streaming nature of the data, which is characterized as continuous, non-stationary, temporally ordered, high-volume, and high-velocity [85, 96]. Further, user feedback arrives at unpredictable rates and order, and is potentially unbounded [296]. In highly dynamic scenarios of real-life recommendations, new items emerge while older items rapidly lose their relevance [138]. In response to these issues, *stream-based* recommender systems have been designed to operate under limited memory constraints, adapt to concept drifts, and provide anytime recommendations almost instantaneously (typically, within 100 ms [138]).

Although the paradigms described in previous sections may capture the temporal and sequential dynamics of recommendations, they do not typically consider the input data as streams [44]. One of the important advantages of stream-based learning over the conventional batch-based learning is the ability of the former to incrementally update the model with new information without having to re-learn the entire model [184]. Learning from evolving data streams entails a trade-off between memory and forgetting [85]. Forgetting mechanisms for stream-based RS have been extensively studied in [184]. Stream-based recommender systems have been proposed both for session-based [96] and matrix-based setups [298, 299].

2.5 Recommendation approaches

A recommender system can be built on the basis of various machine learning techniques and combinations thereof. Because their number is overwhelming, we briefly sketch the state of the art that is relevant to the present thesis.

2.5.1 Content-based filtering

As the name implies, content-based filtering [225] relies on item content features to produce recommendations, which could be textual descriptions, tags, and various attributes (metadata). The content features are used to create a joint representation of items and users in the system. A user vector encoding his/her preferences inferred from the content tokens of previously consumed items is matched to candidate items using a similarity function.

The representation is traditionally based on the vector space model with TF-IDF token weighting [171]. The TF-IDF measure is the product of *term frequency* (TF) and *inverse document frequency* (IDF), which for each token t in item i are defined as follows:

$$TF_{t,i} = \frac{f_{t,i}}{\max_{t'} f_{t',i}}, \quad IDF_{t,i} = \log \frac{|I|}{1 + |\{i \in I : t \in i\}|} \tag{1}$$

where: $f_{t,i}$ is the frequency of occurrences of token t in item i, and I is the item catalog.

The normalized TF-IDF weight is given by:

$$w_{t,i} = \frac{TF_{t,i} \cdot IDF_{t,i}}{\sqrt{\sum_{t'} TF_{t',i}^2 \cdot IDF_{t',i}^2}} \tag{2}$$

The item-to-item or user-to-item affinity can then be computed using, for example, cosine similarity between the weighted token vectors:

$$sim_{u,i} = \frac{\sum_t w_{t,u} \cdot w_{t,i}}{\sqrt{\sum_t w_{t,u}^2} \cdot \sqrt{\sum_t w_{t,i}^2}} \tag{3}$$

Content-based recommender systems have a number of advantages:

- Most importantly, they are able to operate under cold-start and suggest items with no prior interaction history.
- They produce interpretable results, thus increasing system transparency and user trust.
- They can be augmented using the Linked Open Data (LOD) cloud (see the overview of LOD-based recommender systems in [60]).

On the downside,

- They suffer from overspecialization, i.e. producing too obvious results.
- They tend to be less accurate than collaborative filtering (see next sub-section).
- In their basic form, they fail to capture the semantics of items. More recent techniques based on word embeddings can learn such semantic representations [201].

2.5.2 Collaborative filtering

Collaborative filtering (CF) can be viewed as the opposite of content-based filtering, as recommendations are generated solely from user interactions with no reliance on item metadata. Collaborative recommender systems make suggestions of items on the basis of similarities in consumption behavior of users. This approach is considered more powerful than content-based filtering and remains predominant in RS research [113]. It can also provide serendipitous discovery of items, which is difficult with pure content-based methods [222]. However, it is generally less transparent due to "black box" computations, and potentially less secure due to its increased vulnerability to privacy attacks [82]. It also suffers from the cold-start problem.

Because of the abundance of implicit positive-only feedback in real-world applications, *one-class collaborative filtering* (OCCF) methods [215] have been developed to address this type of signals. We only consider this type of CF in the present thesis.

Broadly, CF techniques are divided into *neighborhood-based* (a.k.a. memory-based) and *model-based* ones [59]. The former type utilizes the user-item history directly to make predictions, and is therefore ascribed to lazy learning. A representative algorithm of this type is k-nearest neighbors. In contrast, model-based techniques build mathematical models from historical data to make predictions. Latent factor models, which aim to learn latent features that explain the observed user-item interactions [153], have received wide adoption. Especially popular approaches belonging to this family are matrix factorization and deep neural networks. Apart from these methods, a variety of classification algorithms from machine learning can be employed for building recommendation models. We now briefly introduce some of the popular CF approaches that are relevant to our research.

k-Nearest Neighbors. kNN used to be the de facto standard in collaborative filtering long before the appearance of matrix factorization and deep learning on the RS horizon. This technique remains popular because of its intuitiveness, ease of implementation, and good accuracy. Conceptually, finding like-minded users in CF is very much equivalent to the notion of a user's neighborhood in kNN [10], which makes this technique a natural first choice for building RS. It comes in two variants:

User-user kNN predicts the score of an unknown item i using the preference for this item among k closest neighbors of the target user u. This neighborhood, denoted by $N(u)$, is formed on the basis of similarity between user profiles (rows of the user-item matrix $\mathbf{R}$). The final score for an unobserved user-item pair (u, i) is calculated as follows:

$$s_{\textit{user-user}}(u, i) = \frac{1}{|N_i(u)|} \sum_{v \in N_i(u)} sim(\mathbf{R}_{u,\bullet}, \mathbf{R}_{v,\bullet}) \tag{4}$$

where $N_i(u) \subseteq N(u)$, such that $i \in \mathbf{R}_{v,\bullet}, \forall v \in N_i(u)$.

Item-item *k*NN predicts the score of an unknown item i based on its similarity to the (at most) k-nearest items consumed by the target user u:

$$s_{\textit{item-item}}(u, i) = \frac{1}{|N_u(i)|} \sum_{j \in N_u(i)} sim(\mathbf{R}_{\bullet,i}, \mathbf{R}_{\bullet,j}) \tag{5}$$

The advantage of one kNN variant over another differs from case to case, as it follows from results reported in [5, 221, 300] (in favor of user-user kNN) and [58, 256] (in favor of item-item kNN). Considering the nature of the algorithm, the choice of the method is typically determined by the dimensions of the matrix: user-user kNN is preferred when there are more items that users, and item-item kNN is preferred otherwise.

The choice of a similarity measure plays a key role in this algorithm, since it affects both the computation of neighbors, and the scoring of items [94]. Popular measures include Cosine/adjusted Cosine similarity, Jaccard index, and Pearson correlation.

Recommendations provided by kNN have good explainability because of the intuitive and simple nature of the algorithm. Although the "laziness" of kNN makes it more computationally expensive during the recommendation phase than model-based algorithms, it is customary to pre-compute the nearest neighbors in a training phase to be able to serve near-instantaneous recommendations at prediction time [59]. This also makes kNN naturally suitable for streaming data, since pre-computed item similarities enable recommendations to new users without having to re-train the system. Further, new user-item interactions can be incorporated incrementally by updating the similarities involving only the current item [59]. Recent sequence-aware extensions of kNN have shown very good results on session data [173].

Matrix Factorization. Latent factor models constitute state-of-the-art in batch-operated RS, and have achieved remarkable performance in both rating prediction [154, 242, 250, 278] and top-N recommendation [52, 124, 113, 240] tasks. Particularly effective for the matrix completion task are matrix factorization (MF) techniques, which project users and items to the joint latent space. This is achieved via a singular value decomposition (SVD) of a user-item matrix $\mathbf{R}^{[M \times N]}$, resulting in two low-dimensional embedding matrices $\mathbf{X}^{[M \times K]}$ and $\mathbf{Y}^{[N \times K]}$, $K \ll M, N$.

Each entry in matrix $\mathbf{R}$ can then be approximated via the inner product of the corresponding vectors in the latent space, i.e. $\hat{r}_{ui} = \langle \mathbf{x}_u, \mathbf{y}_i \rangle = \mathbf{x}_u^T \mathbf{y}_i$. The objective of matrix factorization is thus to find the approximation that minimizes the loss function $\mathcal{L}(\mathbf{X}, \mathbf{Y}|\mathbf{R}) = \sum_{u,i}(r_{ui} - \langle \mathbf{x}_u, \mathbf{y}_i \rangle)^2$. If r_{ui} is unobserved, a common strategy is to set $r_{ui} = 0$. In addition, each event can be weighted according to the number of observations, as suggested by Hu et al. [113]. A plausible weighting scheme is $w_{ui} = 1 + \alpha(\#r_{ui})$, where α is a tunable parameter controlling the confidence level increase. This way, the weight matrix $\mathbf{W}^{[M \times N]}$ is obtained. It is also common to add a form of l_2 regularization with a tunable parameter λ to prevent overfitting. The loss function becomes:

$$\begin{aligned} \mathcal{L}(\mathbf{X}, \mathbf{Y}|\mathbf{R}, \mathbf{W}) = \sum_{u,i} w_{ui}(r_{ui} - \langle \mathbf{x}_u, \mathbf{y}_i \rangle)^2 + \\ \lambda(\sum_u ||\mathbf{x}_u||_F^2 + \sum_i ||\mathbf{y}_i||_F^2) \end{aligned} \tag{6}$$

Equation 6 can be minimized via alternating least squares (ALS). This procedure fixes one of the latent vectors and solves for the other vector analytically using ridge regression. The final closed form solutions for both latent vectors are as follows [113]:

$$\begin{aligned}\mathbf{x}_u &= (\mathbf{Y}^T\mathbf{W}_u\mathbf{Y} + \lambda\mathbf{I})^{-1}\mathbf{Y}^T\mathbf{W}_u\mathbf{R}_{u,\bullet} \\ \mathbf{y}_i &= (\mathbf{X}^T\mathbf{W}_i\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{W}_i\mathbf{R}_{\bullet,i}\end{aligned} \tag{7}$$

where: $\mathbf{W}_u^{[N\times N]}$ and $\mathbf{W}_i^{[M\times M]}$ are matrices with respective elements of $\mathbf{W}_{u,\bullet}$ and $\mathbf{W}_{\bullet,i}$ at the diagonal.

The works of Zhao et al. [323, 324] utilize time intervals between interactions to improve the performance of matrix factorization in sequential recommendation settings. Other MF-based recommender systems for sequential tasks have also been proposed [127, 173, 241]. SVD can be adapted for incremental updates via a technique called *folding-in* [258], which allows to inject new vectors to the latent space without re-computing the entire model. This trick helps to improve the scalability of a system, at the expense of prediction accuracy. Vinagre et al. [297] propose a fast incremental MF-based recommender system trained using stochastic gradient decent (SGD), which can handle dynamic streams of positive-only user feedback.

2.5.3 Demographic filtering

The key assumption of demographic filtering is that users with common personal characteristics (gender, age, country, etc.) are likely to share common preferences. Technically, demographic filtering can be implemented using clustering, where each user is assigned to a demographic cluster specific to their profile. The recommendations are then served on the basis of information about other users in the same cluster [224]. Alternatively, demographic filtering can be understood in terms of classification or regression, in which the input variables represent various demographic features of users, and the output values encode their preferences [94]. It is worth noting that these methods do not perform very well on their own, and therefore are usually included as part of more complex hybrid methods [94]. They should also be practised with care because of the potential ethical issues arising from the use of protected attributes (see Section 2.7.6).

2.5.4 Association rules mining

Another classical recommendation technique is *association rules mining* (ARM), whose primary application is market basket analysis in transactional data. Consider a set of sales transactions $T = \{t_1, t_2, \ldots, t_{|T|}\}$, where each transaction represents a shopping basket or a user's purchase history (row of a user-item matrix $\mathbf{R}$). Association rules mining is done in two steps:

1. Frequent itemset generation.
   Mining itemsets can be done using well-known algorithms such as Apriori [4], Eclat [318], and FP-growth [98]. The frequency of an itemset I is determined from its *support*, calculated as follows:
   $$supp(I) = \frac{|\{t : I \subseteq t, t \in T\}|}{|T|} \tag{8}$$
2. Rule generation.
   Each rule is a binary partitioning of a frequent itemset, taking the form $X \Rightarrow Y$. The acceptance of a rule is determined by some threshold on its *confidence*, calculated as follows:
   $$conf(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)} \tag{9}$$

Using this method, one can build a simple recommender system by mining frequent itemsets from the rows of matrix $\mathbf{R}$, and then extracting all the rules "supported" by a given user and satisfying minimum confidence. Thus, the antecedent of each extracted rule contains one or more of the user's known items, whereas the consequent of a rule contains recommendable items. The aggregated confidence of rules can then be used to determine the final ranking of candidate items. The choice of thresholds for the support and confidence of rules has direct effect on the coverage and the accuracy of recommendations. The ARM technique has been successfully applied to clickstream data with notable improvements over kNN methods in terms of scalability, accuracy, and coverage [194].

Because the method considers each transaction as a basket of items, the sequence of user interactions is not preserved. To overcome this limitation, *sequential pattern mining* (SPM) and *contiguous sequential pattern mining* (CSPM) methods aim to uncover the behavioral patterns of consumption in which the ordering of events is taken into consideration [232]. The latter variant is the most restrictive one, since it imposes the adjacency of items. Sequential pattern mining is often used for the next-item prediction task in session data [173, 195, 202, 315]. A choice between ARM, SPM and CSPM is typically driven by the importance of the ordering of events in a given application domain.

2.5.5 Bandit algorithms

Reinforcement learning problems address one important characteristic of real-life RS that is difficult to achieve with traditional batch-oriented methods, namely the need for adaptability [94]. Indeed, deployed recommender systems must be able to reinforce themselves with new observations to effectively handle cold-start and ever-changing context. One popular class of reinforcement learning problems is the multi-arm bandit problem, see Section 2.3.3.

In the most common scenario of a stochastic multi-arm bandit with a set of k arms and Bernoulli rewards $r \in \{0, 1\}$, the recommendation agent at each time step t seeks to pull an arm $i(t) \in \{1, \ldots, k\}$ that maximizes the expected total reward after T steps, i.e. $r_T = \mathbb{E}\left[\sum_{t=1}^{T} r_{i(t)}\right]$. The choice of an arm $i(t)$ at each time step is governed by a bandit policy, given the knowledge about the number of pulls $n_i(t)$ and the number of rewards $r_i(t)$ accumulated by each arm i up to time t. For example, it could be one of the following policies:

ϵ-greedy. Based on a pre-defined probability threshold $0 < \epsilon < 1$ and a random sample $p(t) \sim \mathcal{U}(0, 1)$, the policy picks an arm greedily as follows:

$$i(t) = \underset{i=1,\ldots,k}{\arg\max} \begin{cases} \frac{r_i(t)}{n_i(t)}, & \text{if } p(t) > \epsilon \\ p(t), & \text{otherwise} \end{cases}$$

The exploration parameter ϵ can be adapted to shrink over time at a certain rate in order to reduce the amount of exploration, e.g. $\epsilon_t = c/t, 0 < c < 1$ [266].

Upper Confidence Bound (UCB). This policy follows the principle of "optimism in the face of uncertainty". For each of the first k steps, UCB explores a new arm. After that, it chooses an arm deterministically such that:

$$i(t) = \underset{i=1,\ldots,k}{\arg\max} \left[\frac{r_i(t)}{n_i(t)} + \sqrt{\frac{2 \ln t}{n_i(t)}} \right]$$

The above equation corresponds to UCB1 [14], which is the most basic variant of the algorithm. The second term of the equation encodes the approximation for "optimism" by considering less explored arms as more uncertain [38]. By extending this term, more sophisticated variants of the algorithm have been proposed, such as UCB2 [14], UCB-tuned [14], and MOSS [13].

Thompson Sampling (TS). This policy implements the "randomized probability matching" strategy [263]. The choice of an arm depends on its probability of being optimal, which is sampled from the Beta posterior distribution:

$$i(t) = \underset{i=1,\ldots,k}{\arg\max}\, \theta_i(t) \sim Beta(r_i(t) + 1, n_i(t) - r_i(t) + 1)$$

The two parameters of Beta distribution hold the number of successes and the number of failures of each arm. This simple yet powerful Bayesian approach has shown excellent performance in addressing complex online problems [91].

To make bandit algorithms suitable for the top-N recommendation problem, a few approaches exist. One possible solution is to assign a separate bandit to each recommendation slot of a top-N list. This approach is taken in *ranked* [158, 235] and *independent* [149] bandits. The drawback of this method is that it does not fully utilize all the available feedback, and hence takes longer to converge than a single bandit solution. A way to address this problem is to allow a single bandit to pull several arms at each round in order to construct their ranking. This type of MAB is known as a *multiple-play bandit* and has been studied in [150, 172]. Another emerging approach is based on ensemble learning, where the bandit algorithm decides which recommendation model to choose for filling the slots in the top-N ranking by alternating between exploiting the currently best model and exploring the potential of other models [77, 281].

2.5.6 Hybrid/ensemble methods

Oftentimes, best results can be achieved when different algorithms are either combined in a single model or complement each other to reach a consensus. The success of these approaches has been witnessed in recommendation competitions such as the Netflix Prize contest and KDD cups [281].

The first approach is usually referred to as *hybrid* RS. Many recommender systems belonging to this category take advantage of heterogeneous data sources, such as content features, collaborative signals, and user demographics, in order to derive richer representations for modeling user behavior. For example, Pazzani [224] introduces a method called "collaboration via content" to address the sparsity of user-item interactions. The proposed hybrid approach has proved more successful at making recommendations than purely collaborative or content-based filtering. In a similar fashion, these two filtering methods have been combined to produce a movie recommender [251]. Hybridization is also a common way of tackling cold-start in RS [7, 53, 222, 260]. One of the well-established techniques that enables easy blending of various data sources is a factorization machine [239]. It was used in [90] to combine user and item characteristics with behavioral data to build a hybrid recommender system for e-commerce. Other successful hybrids have been designed by combining matrix factorization with neighborhood models [73, 151] and bandit algorithms [136, 326]. A fusion of user feedback types - explicit and implicit - has also proved effective [152, 162].

The second approach is known as *ensemble* RS. The idea behind this method is to train a number of component recommenders, whose individual outputs are aggregated for the final prediction. The aggregation is typically done by taking the average or the (weighted) majority vote [281]. This allows to reduce the variance of learning algorithms on unseen data. Ensembles are often much more accurate than the individual predictors that they are comprised of [64]. One of the most celebrated examples of this is the recommendation ensemble of BellKor's Pragramatic Chaos team that won the grand prize of the Netflix contest [9]. Other successful examples in the RS practice include ensembles of matrix factorizations [56, 309], multi-arm bandits [281], and mixed predictors [16, 118, 224].

2.6 Evaluation

We now review commonly used methods, metrics and criteria for evaluating top-N recommender systems.

2.6.1 Methods

A typical RS evaluation pipeline consists of an offline evaluation based on some pre-recorded data, followed by an online evaluation on real users. This last phase is usually performed when there is sufficient evidence that the recommender outperforms some important baselines on recorded data. In academic research, online evaluation is difficult to perform and not generally practiced for evaluating proof-of-concept prototypes. In industry, however, A/B testing is widely used for online evaluation.

The offline approach does not involve any real interaction with RS users, but has the advantage of being easy to conduct and reproduce via publicly accessible datasets and standardized protocols and metrics. This type of evaluation can be done in either batch or sequential (streaming) mode. We outline each of these methods next.

Batch mode The batch mode of evaluation remains predominant in RS research. Its central premise is to make all predictions at once. With this method, the total event history is segregated into distinct subsets, where the largest one is used for training the model (training set), and the smaller one is held out for the evaluation (test set). Often, a separate validation set is used for hyperparameter tuning. Depending on the data at hand, there are two basic ways of sampling user-item pairs:

Random sampling. Usually, the separation of the sets is done by a *random* sampling of a certain percentage of interactions from the original dataset to build the test set. This proportion typically ranges from 10% to 50%. The remaining events form the training set. By repeating this procedure k times while ensuring that each event is seen only once in the test set, we arrive at the k-fold *cross-validation* method. This procedure helps to reduce the variance of the estimates.

Chronological sampling. Dataset splitting can also be done chronologically, when timestamps of events are available in the data. According to this strategy, the experimenter chooses a time instance, which divides user profiles into past and future events. With this setup, the recommender is trained on past data to predict future data, which clearly leads to a more realistic evaluation. Although classical cross-validation in this case is not feasible, re-sampling and time-windowing methods can be employed to a similar effect [40].

Apart from the popular proportion-based splitting strategy, there exist protocols that fix either the number of known items, or the number of withheld items per test user for evaluation. They are referred to as "given n" or "all but n" protocols [95].

Sequential mode Unlike the batch mode of evaluation, the sequential method reveals test events to the experimenter sequentially, and hence measures the RS performance in an incremental fashion. This closely mimics the real recommendation process, where each user action triggers a recommendation request. For example, the user's browsing behavior on an e-commerce website may trigger recommendations of substitute, complementary, and discounted products [119] (aiming at up-, cross-, and down-selling, respectively), depending on the context.

In principle, dataset partitioning is not required in sequential mode of evaluation [296], if online learning is adopted. Nevertheless, some historical data is typically provided to pre-train the model prior to entering the prediction phase. For sequence-aware RS, the dataset partitioning can be done either *between-* or *within-*sequences [232]. The former procedure assigns complete sequences to training or test sets, whereas the latter allows splitting a sequence into contiguous subsequences.

The most common sequential protocol for streaming environments is known as *prequential evaluation* [85, 296]. It consists of *test-then-train* operation cycles that run on sequentially arriving user-item pairs (Algorithm 1).

Algorithm 1 Prequential evaluation

```
1: for each user-item pair (u, i) arriving from the stream do
2:     Produce top-N recommendations for user u
3:     Evaluate recommendations against item i
4:     Update model with new observation (u, i)        ▷ optionally
```

This protocol corresponds to the next-item recommendation task, which is the special case of the "best-item" recommendation objective mentioned in Section 2.1. The advantage of this evaluation method over the batch mode is the ability to continuously monitor the system's performance over time against several metrics simultaneously [296]. The protocol naturally suits the multi-arm bandit setting (and reinforcement learning in general), but can also be used for traditional matrix completion problems (e.g. [12, 297, 298]). However, this protocol is strict in that it only evaluates the prediction of the immediate next event, while ignoring the subsequent ones [296]. To relax this, *sequential evaluation with look-ahead* [232] matches recommendations against next n items of the current user (only possible offline). Alternatively, the current observation can be matched not only with the current recommendations, but also with a set of previous ones [296].

A/B testing Different from the previously discussed protocols, A/B testing is an *online* evaluation method, which is widely used for evaluating new algorithms, design features, or even entire websites. It can be viewed as a two-sample hypothesis testing methodology, where the null hypothesis represents the control entity ("A") and the alternative hypothesis represents the treatment entity ("B"). For a recommender system, the null hypothesis is associated with the current state of RS, while the alternative hypothesis typically represents a new algorithm, a modification of an existing algorithm, a new layout, etc. The user traffic is then randomly divided between variants A and B, and the RS is evaluated against some metric for some period of time. By measuring the difference between A and B, the decision on the acceptance of the alternative hypothesis is taken. The vast majority of online platforms, such as Amazon and eBay, run online experiments on a continuous basis [231] to evaluate the effectiveness of the system with real users.

2.6.2 Metrics

The accuracy of a top-N recommender system is usually assessed not in absolute rewards, but in relative measures taking values from 0 to 1 (the higher the better). To evaluate an N-sized recommendation list $L(q)$ generated for request q, we calculate rewards or *true positives* (TP) by comparing recommendations $L(q)$ to the test set $T(q)$ containing the hidden ground truth items for the active user (or session) that issued the request. This comparison also gives us the related counts of *false positives* (FP), *true negatives* (TN), and *false negatives* (FN), summarized in the contingency table of Figure 1.

		CONSUMED	
		✓	✗
RECOMMENDED	✓	TP	FP
	✗	FN	TN

Figure 1: Contingency table

By constructing contingency tables for the whole set Q of recommendation requests, we can compute useful retrieval metrics, such as *precision* and *recall* at the desired cutoff N:

$$Precision@N = \frac{1}{|Q|} \sum_{q \in Q} \frac{TP_q}{TP_q + FP_q} \tag{10}$$

$$Recall@N = \frac{1}{|Q|} \sum_{q \in Q} \frac{TP_q}{TP_q + FN_q} \tag{11}$$

These metrics show how many rewards we get in relation to the size of the recommendation list (precision), or the size of the test set (recall). Because precision and recall trade-off against each other, they are often reported via their balanced harmonic mean, known as the *F1-measure*:

$$F1@N = \frac{1}{|Q|} \sum_{q \in Q} \frac{2 \cdot Precision_q@N \cdot Recall_q@N}{Precision_q@N + Recall_q@N} \tag{12}$$

It is a known fact that top-N rankings introduce a position bias, according to which the probability of a click on an item decays with its rank in the list [50, 95]. The above evaluation metrics cannot accommodate this bias. For this reason, top-N recommender systems are often evaluated using ranking measures. One of them is *mean average precision* (MAP), which takes the arithmetic mean of precision at all available recall levels:

$$MAP@N = \frac{1}{|Q|} \sum_{q=1}^{|Q|} \frac{1}{|T(q)|} \sum_{i=1}^{N} Precision_q@i \tag{13}$$

where: $Precision_q@i = 0$ if $L_i(q) \notin T(q)$.

Another popular ranking measure is *normalized discounted cumulative gain* (NDCG). Denoting $r_i \in \{0, 1\}$ as the reward obtained from displaying an item at rank i of $L(q)$, NDCG can be computed as follows:

$$NDCG@N = \frac{1}{|Q|} \sum_{q=1}^{|Q|} \frac{DCG_q}{IDCG_q}$$

$$\text{where: } DCG_q = \sum_{i=1}^{N} \frac{r_i}{\log_2(i+1)}, \quad IDCG_q = \sum_{i=1}^{\min(N,|T(q)|)} \frac{1}{\log_2(i+1)} \tag{14}$$

A ranking measure that is often used for evaluating next-item predictions is *mean reciprocal rank* (MRR). It averages the reciprocals of the first (i.e. topmost) relevant item's rank across recommendation lists:

$$MRR@N = \frac{1}{|Q|} \sum_{q=1}^{|Q|} \frac{1}{rank_q@N} \tag{15}$$

where: $rank_q@N$ refers to the position of the first relevant item in top-N recommendation list $L(q)$.

Despite their widespread use, the illustrated evaluation metrics represent only a small fraction of available metrics in the arsenal of RS research, and have been selected because of their relevance to the present thesis and their popularity in the field. To ensure the usefulness of evaluation, the set of metrics has to be carefully chosen in accordance with the specific objectives that have been set for the concrete RS usage scenario and the characteristics of its domain [261]. These objectives also determine other evaluation criteria beyond accuracy that might be important to consider in a particular use case.

2.6.3 Criteria

Recommender systems can be assessed for a variety of criteria (or properties), which have been covered in detail in [95]. Some of the most important ones are outlined below.

Accuracy. This is the key property that recommender systems are being assessed for. It can materialize in various measures depending on the application domain, e.g. click-through-rate in advertising, conversion rate in e-commerce, reciprocal liking in dating, and so on.

Utility. This criterion can be defined as the value that RS stakeholders (i.e. users, providers, or platform owners) gain from recommendations. In the e-commerce domain, for example, recommendations serving some business-critical utility such as revenues or customer retention may be even more important than accurate predictions of purchases.

Coverage. Real-world datasets suffer from the *long-tail effect*, according to which the largest part of the product catalog is rarely or never consumed. Therefore, optimizing a recommender for a wider catalog coverage with the inclusion of long-tail items is often desirable. Likewise, the ability of generating accurate recommendations for as many users as possible is another desired property.

Explainability. The reliability of recommendations is not obvious to a user unless they can be reasonably justified. This may affect the usability of the entire recommendation module on a website. It also raises wider ethical concerns around algorithmic opacity and behavior manipulation.

Novelty/Serendipity. It is important to ensure that the system suggests relevant items beyond those that the user has already seen or would have found anyway. Serendipitous recommendations are those that are unexpected in the context of the request, yet still interesting to the user.

Diversity. Generating a diverse list of suggested items can be desired to avoid the effect of a "filter bubble" in recommendations. Besides, displaying only similar items in recommendations is likely to reduce the usefulness of recommendations. Diversity is also closely related to the issues of fairness, since dissimilar items can help to mitigate potential discrimination.

In many practical situations, it is desirable to jointly optimize multiple properties of a recommender system, which are not necessarily well-aligned. For instance, increasing the diversity, coverage or novelty of recommendations may reduce their accuracy. To this end, multi-objective learning in RS is often employed [158, 244, 304].

2.7 Ethics in recommender systems

2.7.1 Law, ethics, and morality

As the societal impact and the degree of algorithmic sophistication of recommender systems continue to increase, it is worth studying these systems from an ethical perspective. Obviously, RS practitioners have an obligation to adhere to policies prescribed by legislation. However, laws alone are not sufficient to protect users from the wide range of undesirable side effects of using recommender systems and other decision-making machinery. Unlike legislation, which has clear political bounds, ethics aim to protect core (or *intrinsic*) values shared by all humans regardless of their cultural, political, or geographical borders. Among these values are autonomy, freedom, security, knowledge, and resources [199]. Understanding the difference between law and ethics is, in essence, knowing to distinguish between what one has a right to do (legally, that is), and what is right to do [49].

The discipline of *ethics* is concerned with studying *morality*, which can be defined as "a system of rules for guiding human conduct, and principles for evaluating those rules" [283]. The purpose of this system is "to prevent harms and evils" [89] and "to promote human flourishing" [283]. Narrowing this down to the field of RS, we can define recommendation ethics as *the study of the moral system of norms for serving recommendations of products and services to end users*, see also [217].

2.7.2 Recommenders as moral agents

The motivation for recommendation ethics as a field of study comes from the realization that recommender systems represent agents that act autonomously on behalf of the respective bodies who deploy them, thus inheriting their moral agency. Therefore, at least theoretically, RS can be thought of as a type of *artificial moral agents* [301]. Such agents share certain moral competencies and abilities with human decision makers [111]. Whereas they may not be entirely equivalent to human moral agents, we claim that they can have "functional morality" by satisfying two criteria: (a) autonomy, and (b) sensitivity to ethical values [301]. Furthermore, autonomous recommendation agents can be regarded as moral agents because they are (a) sources of moral action, and (b) can cause moral harm or moral good [81]. In terms of Moor's framework [200], the current state of (the very few) ethics-aware recommender systems ascribes them to either "ethical impact agents", or "implicit ethical agents". The former type is the weakest sense of a moral agent, which merely has ethical consequences to its actions. Numerous examples of such consequences are provided in [217]. The latter type of agent employs basic ethical intelligence, which is prescribed by some in-built moral rules and principles. There is evidence that such rule-based implicit ethical agents have begun to appear on the RS horizon [270, 282].

2.7.3 Moral responsibility

Being morally responsible for taking an action, such as giving recommendations, is conditioned on two criteria: *causality* and *intent* [283].

The principle of *causality* implies that some agent A is held responsible for action X if A caused X, regardless of the intent. For example, consider a recommendation agent being trained on historically biased data, resulting in discriminatory results (e.g. a dating agent that learns to prevent African Americans from getting recommendations of partners that belong to a different race [293]). Even though the developers of such an agent did not have any intent to discriminate, they are nevertheless morally responsible for the outcome.

On the other hand, an agent can also be held responsible for the mere fact of immoral *intent*, even if the actual harm has been avoided. For instance, introducing first-degree price discrimination in a recommender system is commonly perceived by the general public as unethical [287] (it can also be illegal in many cases under the Robinson-Patman Act [70]). Even if such recommendations go unnoticed, the designers of the system are morally culpable because of the system's intentional disparity of treatment.

2.7.4 Principles of accountable algorithms

Moral responsibility is part of a broader "robust and intuitive notion of accountability" [208], which means that an agent, or a group of agents are *answerable* for a caused harm [209]. According to FATML, the five principles of accountable algorithms are as follows [62]:

Responsibility. Reveal possible ways of redress for adverse individual or societal impacts of an algorithm-driven decision mechanism, and designate an internal body responsible for the timely settlement of such issues.

Explainability. Provide comprehensible, even to non-technical users and stakeholders, explanations of data and algorithms employed in automated decision making.

Accuracy. Identify, record, and make available all sources of error and uncertainty in data and algorithms, which can be used to understand their implications and frame mitigation strategies.

Auditability. Disclose necessary information in the form of documentation, API, or terms of use, which would enable interested third parties to perform monitoring, checking, and criticism of the algorithm's operation.

Fairness. Take necessary steps to prevent discriminatory or unfair algorithmic decisions with respect to various demographic groups (race, gender, etc.).

These principles make it easier for RS developers to reason about the moral accountability of their artifacts as they proceed through various development stages.

2.7.5 Ethical impact

Given that the field of recommendation ethics is still in its infancy, there is a need for identifying and categorizing all sorts of implications of recommender systems on the society, which would help to move the field further towards ethics-awareness. To this end, the recent study by

Milano et al. [191] presents a two-dimensional taxonomy of the ethical impact of recommender systems, which we briefly summarize below.

Along the first dimension, the impact can manifest itself as:

1. A loss in *utility* of recommendations, e.g. irrelevant recommendations for users, reduced revenues for providers.

2. A violation of *rights*, e.g. various forms of unfairness.

Along the second dimension, the impact can constitute:

1. An *immediate* harm to stakeholders, e.g. generating unfair recommendations.

2. An exposure to *future* risks, e.g. privacy breaches.

The assessment of the ethical impact of RS must be done in consideration of all the affected parties. This is especially relevant in multi-sided markets bringing together multiple types of participants (e.g. e-commerce auctions, social networks, dating platforms, ride-sharing services, etc.). These types include RS users, providers, platform, and society as a whole. Refer to Milano et al. [190] for a detailed discussion around the ethics of multi-stakeholder recommender systems.

2.7.6 Algorithmic fairness

The subject of algorithmic fairness has drawn considerable attention in recent years, currently enumerating more than 20 different formalizations adapted from the social science literature [294]. This goes to show that no universal definition of fairness exists, and that it is highly contextual [186]. The ultimate goal of fairness is to enable non-discriminatory predictions with machine learning. Discrimination can manifest itself in either a *direct* or *indirect* way [321]. The former type takes place when sensitive (i.e. protected by law) attributes are explicitly used in generating non-favorable decisions for the users. This causes the effect of a *disparate treatment* [84] of users with respect to the protected attribute. The latter type of discrimination occurs when a seemingly neutral attribute of users results in non-favorable decisions because of its correlation with a sensitive attribute, leading to a *disparate impact* [84]. Unfairness is typically a property of biased data that contains human-induced prejudice and stereotypes, and discriminatory model failing to account for sensitive information and class imbalances.

The most widely used definitions of fairness fall under two main categories [186]:

Group fairness. A predictor satisfies group fairness if a given group-conditional quality metric is comparable for individuals across protected groups [84, 271]. Depending on the choice of the quality metric, different notions of group fairness have emerged, such as *demographic parity* [68], *equality of opportunity* [100], *equalized odds* [100], *calibration within groups* [141].

Individual fairness. A predictor satisfies individual fairness if it produces similar outputs for similar individuals [84]. The key aspect of individual fairness is the choice of a distance metric that determines the similarity between individuals (without considering protected attributes). Some of the known formalizations that belong to this category are *fairness through awareness* [68] / *unawareness* [155], *calibrated fairness* [169], *counterfactual fairness* [155].

Addressing algorithmic fairness in practice can be surprisingly challenging [186], given the multitude of existing definitions and the fact that some of them are mutually exclusive [84]. Translating fairness formalizations to recommendation scenarios is further complicated by the immense sparsity of relevance evaluations and the multi-stakeholder nature of recommender systems [72]. To this end, Burke [34] introduces the notion of *multi-sided fairness*, which takes into account the interests of recommendation consumers, providers, or both parties (denoted as *C*-, *P*-, and *CP*-fairness, respectively). Most often, this type of fairness concerns reciprocal recommendations, e.g. in dating [327] or job matching [193]. The growing body of work on the fairness of rankings [24, 265, 313, 314] finds practical applicability in top-N recommender systems. A particularly useful concept is that of re-ranking [132, 168] — a post-prediction procedure aiming to rebalance the ranked outputs to satisfy some fairness criteria. In many cases, this can be achieved with minor compromises in prediction accuracy [168, 265, 314].

3 METHODOLOGY

No amount of experimentation can ever prove me right; a single experiment can prove me wrong.
- Albert Einstein

This section details the methodological approach that has been followed in the thesis.

3.1 Collaboration with Apptus Technologies

Through our collaboration with *Apptus Technologies*[8] - a Nordic leader in e-commerce merchandising - we were able to get access to real-world shopping session data to conduct part of our research. At the time when our collaboration was initiated (early 2014), public datasets with e-commerce session data were virtually non-existent, and their availability today is still limited (4 such datasets are reported in [232]). The contribution of Apptus Technologies to the research presented in this thesis is threefold:

- Two datasets containing shopping sessions from *books* and *fashion* domains were provided for experimentation. Both datasets are used in Paper I and Paper III.
- The Apptus' *eSales* development platform was provided for building prototypes and running simulations.
- The joint research on a session-based recommender system was conducted, whose results are reported in Paper III.

To comply with the ethical policies, the analysis of proprietary datasets was performed on company premises. This was done to eliminate the risk of any accidental leakage of data belonging to the client base of Apptus Technologies. Furthermore, all session data were anonymized to protect the personal privacy of data subjects.

[8] **http://www.apptus.com**

3.2 Choice of methods

To answer the research questions of this thesis, we apply five research methods: (1) *experiment*, (2) *design science*, (3) *survey*, (4) *case study*, and (5) *literature review*. The connection of these methods to the research questions (and implicitly to the corresponding papers) is shown in Table 1.

Table 1: Research methods

Research method	Research question(s)
Experiment	I (a), III (a), III (b), III (c), IV (a), V (b)
Design science	III, III (b), III (c), IV, IV (a), V (a), V (b)
Survey	I (b), II (a)
Case study	I, V
Literature review	II, (V)[9]

The appropriateness of each method is assessed on the basis of two criteria defined by Oates [211]:

- the method enables the researcher to find an answer to the research question.
- the method enables the researcher to present the evidence and the conclusion, and to argue convincingly that a new knowledge has been created.

As follows from Table 1, each research paper applies more than one research method. This is necessary to meet the multiple objectives of each paper. In most cases, one of the employed methods is used to evaluate the results of the main method, e.g. a design science followed by an experiment (Paper III and Paper IV), or a literature review followed by a feasibility survey (Paper II). In Paper I, two different methods are used in *triangulation*, i.e. providing independent perspectives on the same issue.

[9] secondary method

Next, we present each of the methods separately and give more details on how they have been applied to our research.

3.3 Experiment

Experiment is a research method that explores cause and effect relationships, aiming to prove or falsify a hypothesis about the link between a factor and an observed outcome [211]. To formulate a hypothesis, a causal factor (independent variable) and an observed/measured quantity (dependent variable) need to be identified. In our research, the experimentation strategy lies in determining how, and to what extent, the use of a certain recommendation algorithm (or some modification thereof) affects the pre-defined evaluation measure(s).

A specific type of experimentation we employ in this thesis is a *simulation experiment*. The design of such an experiment involves the construction of a simulation model (or simply a simulator) in a way that closely mimics a real-world system [116]. The simulations models used in our research employ either batch evaluation (Papers I & V) or streaming evaluation (Papers III & IV) protocols. In these studies, an experimental unit represents the result obtained from a single run or a batch, which are later averaged or aggregated to yield an output value [252]. Since experiments cannot prove anything with absolute certainty, they are typically repeated multiple times, under varying conditions [211]. In our streaming simulator that is used in Paper III, the aggregated result is obtained after averaging the outputs of 10 simulation runs with different random seeds, due to the randomized nature of an algorithm. In Paper IV, this has not been necessary as the proposed algorithm is deterministic[10]. The varying conditions have been supplied by means of three real-world datasets. The above methods comply with the standard practice for evaluating data streams [25]. In our batch simulators, the repetition is achieved by performing a 5-fold (Paper I) and a 21-fold (Paper V) cross-validation.

In our research, experiments serve three purposes. First, they are used as an empirical proof of concept after the *design science* step, where

[10]The evaluation framework itself (`FlowRec`) has support for stream restarts

the new prototype is evaluated against alternative baselines or configurations (Papers III, IV, & V). Second, the simulator can be part of the artifact itself, as is the case with Paper IV. Here, a simulation experiment is performed to showcase its own functionality. Third, a simulation experiment can also be part of a mixed methodology, such as triangulation. It allows to capture a more holistic and complete portrayal of the case under study. In Paper I, the results of experiments are triangulated with the results of the survey to answer the main research question (RQ I).

3.4 Design science

Design science aims to build technological artifacts that serve human purposes [178]. More specifically, it seeks to "create innovations that define the ideas, practices, technical capabilities, and products through which the analysis, design, implementation, and use of information systems can be effectively and efficiently accomplished" [106].

Design science artifacts can be of four types: *constructs* (vocabulary and symbols), *models* (abstractons and representations), *methods* (algorithms and practices), and *instantiations* (implementations and prototypes) [107]. The artifacts produced in our research belong to the last three types, i.e.:

Models. This type of artifact is encountered in Paper V. After conceiving the idea of *preferential fairness* and putting it in the context of existing fairness definitions, the proposed notion is formalized as a model.

Methods. This type of artifact is seen in Papers III, IV, & V, whose core ideas take the form of algorithms expressed either mathematically or in pseudo-code.

Instantiations. All the proposed algorithms have been implemented to produce working prototypes, i.e. *instantiations*. The prototype of Paper IV has been released as an open-source Python library.

Design science is essentially a problem-solving paradigm [106, 211].

Its research cycle iterates between five steps [291], which we briefly outline below.

1. **Awareness**. This step gives birth to a research problem, identified in relevant practice or research. The motivations and intuitions for our research ideas are summarized in the thesis' introduction (Section 1).

2. **Suggestion**. This is a creative step, where a novel solution or a tentative idea is envisioned for a problem at hand. In this phase, we make important design choices, such as utilizing Thompson Sampling for the bandit ensemble of Paper III, or choosing `Scikit-Multiflow` as the basis for our streaming framework proposed in Paper IV.

3. **Development**. In this phase, the idea is implemented according to the type of the artifact. All the proposed algorithms are first pseudo-coded, and then materialized into functional prototypes.

4. **Evaluation**. This step assesses the artifact on some implicit or explicit criteria outlined in the first step. It usually reveals deviations from expected results, in which case the newly gained insights are communicated to another round of Suggestion (step 2). This iterative workflow has been used to incrementally improve our prototypes (at the very least, by tuning hyperparameters). We have followed the "proof-of-concept" evaluations, whose purpose is to demonstrate that the proposed solution has certain properties, or that it behaves in a particular way under certain conditions [211].

5. **Conclusion**. The research cycle ends when the results are perceived as satisfactory and "firm", meaning that the observed behavior can be easily re-invoked if needed, and any anomalous behavior serves as ground for further research.

3.5 Survey

The survey methodology has three main characteristics [228]:

- It is used to obtain specific quantitative descriptions of the population in a standardized way.
- It collects information from people by asking them structured and pre-defined questions.
- It focuses on a fraction of the population - a *sample*, whose size is large enough to enable statistical analyses.

This methodology is employed in Paper I and Paper II. Our design choices in each case are guided by the survey's objective. In Paper I, the goal of the survey is to reveal important characteristics of commercial implementations of recommender systems. Thus, the data collected through the survey represents *facts* about the studied phenomenon. In Paper II, the survey is used to study the feasibility of the proposed "ethical toolbox" for RS users. Here, the collected data captures user *opinions* on specific ethical issues. In both cases, web-based questionnaires are employed as a data generation method. This method allows to reach many people across the globe quickly and at a low cost [211]. Both studies are cross-sectional, i.e. focusing on a particular "snapshot" of a population in time [48].

Other important design considerations have to do with the choice of sample frame and size, as well as sampling technique. In Paper I, our sample frame includes numerous e-commerce platforms that feature recommendation engines. Because of the anticipated difficulties in obtaining business-sensitive insights into the internal workings of proprietary software, we aim for meeting the minimally acceptable threshold of 30 companies as a final sample size. This is considered the de facto standard for studies with low estimated response rates [211]. The goal has been reached with 31 total responses and a response rate of 12.4%. We use random sampling as our sampling technique. The online questionnaire has been communicated to the respective development teams via email or private chat session.

The target group of the survey in Paper II is individuals rather than companies. The questionnaire has been disseminated through Facebook groups of numerous European universities. Since acquiring people's opinions is clearly an easier task than surveying companies, we opt for a larger sample, which has yielded 224 responses. We narrow our sample frame to students and faculty members, and only those who have confirmed their experience of using recommender systems. This sampling technique is known as self-selection sampling [211]. The rationale for choosing this method is to improve the validity of results through increased confidence level of the respondents.

3.6 Case study

A case study allows a researcher to closely investigate a contemporary phenomenon within its natural context [71, 248]. This investigation can be of different types, e.g. exploratory, descriptive, or explanatory [211]. Case studies are flexible in that they allow for a study design that suits a particular case, thus fully capturing its complexity [71]. The methodology is encountered twice in the thesis.

In Paper I, we conduct a *comparative* case study, which focuses on two or more instances of specific phenomena [192]. Its goal is to unearth contrasts and similarity patterns across the examined cases. Specifically, we compare several well-known recommendation algorithms for their predictive performance in a lab environment, and in parallel compare their usage dynamics in an industrial sector by running a survey. Hence, the unit of analysis is an algorithm. Such practice of employing different research methods within a case study is not uncommon [248].

In contrast with the above approach, where the units of study are off-the-shelf algorithms, the case study of Paper V is primarily of an exploratory nature. That is, it needs to understand and devise its units of study (namely, the measure of fairness and the re-ranking methods) based on the provided context and existing literature, before moving on with their development and analysis.

Both case studies also contain an explanatory counterpart, which seeks to explain the observed outcomes. Ultimately, their insights are meant to be "a step to action", i.e. directly interpretable and applicable to real-world scenarios [48].

Case studies can be summarized using the following simple 3-level hierarchy: *context* $\rightarrow$ *case* $\rightarrow$ *unit of analysis* [71]. This is done in Table 2.

Table 2: Case studies of the thesis

	Paper I	**Paper V**
Context	E-commerce platforms	Speed dating
Case	RS in industry	Fairness of RS
Unit of analysis	RS algorithms	Preferential fairness, re-ranking methods

3.7 Literature review

Literature review is the necessary means to provide the foundation and the conceptual framework for a research [211]. It is used as a secondary research method in each paper of the thesis to identify gaps in the literature and motivate the formulated research goals. In Paper II, it is also used as a primary research method to address the ethical challenges in recommender systems.

In this thesis, we adopt a rigorous, evidence-focused literature review methodology proposed by Hagen-Zanker & Malett [97]. This method follows the key principles of a systematic literature review (rigor, transparency, and replicability), at the same time enabling a more flexible retrieval and analysis mechanism. This mechanism entails three separate activities:

1. **Database search** using pre-defined search strings. As suggested by Wohlin [307], we initiate our search using Google Scholar[11] to alleviate the bias towards any specific publisher.

2. **Snowballing**, which is applied to the initial literature selection identified in the previous step to drive the study further. We perform snowballing by systematically examining both the references list of the paper, and the citations to the paper. These

[11] **https://scholar.google.se/**

techniques are known as backward and forward snowballing, respectively [307]. Snowballing is very useful for identifying influential, seminal work within the field of study [97].

3. **Grey literature capture**. Relevant material can often be found in sources beyond traditional peer-reviewed publications, such as academic journals and conferences. The inclusion of the so-called "grey literature" sources (e.g. working papers, reputable blogs, technical reports, etc.) makes it possible to find relevant material that has just been released and cannot be picked up via snowballing. It contributes to the breadth, relevance, topicality, and utility of a literature review [97].

In Paper II, the above methodology has been applied to each of the RS development stages. At each stage, the literature review allows us to reveal existing ethical issues and known countermeasures. Being evidence-focused, our literature study largely relies on the identification of existing examples of moral dilemmas in the RS practice to support the narrative. The findings of the literature review have been summarized in the form of an ethical recommendation framework presented in Paper II.

3.8 Research limitations

In practice, the application of the chosen research methods entails certain limitations and threats to validity that are worth acknowledging.

3.8.1 Experiments

Our experiments are limited to offline simulations on recorded data. Despite the predominance of this evaluation method in research, its results cannot adequately measure true user satisfaction [19]. To reliably measure how an algorithm influences users' behavior in reality, it has to be tested online (e.g. through A/B testing). Whereas our collaboration with Apptus Technologies could, in theory, enable us to perform A/B testing in Papers I & III, this evaluation has not been practically possible.

The absence of statistical significance testing in some of the performed experiments could impact their internal validity. To increase validity, we use both the *methodological* triangulation and *source* triangulation [248] in Paper I. The former intersects the results obtained from different methods, whereas the latter employs several distinct data sources for evaluation. We also employ a 5-fold cross-validation to reduce the effect of variation in the training data. In Paper V, we further increase the number of folds to 21 to obtain more reliable average estimates. Besides, the comparative evaluation of (re-ranking) algorithms is of minor importance to Paper V. In streaming data, cross-validation is not applicable [85]. In paper Paper IV, we employ source triangulation (using 3 datasets), which in this particular case is more relevant than significance testing given that the evaluated algorithms (except for the ensemble learner) are deterministic.

3.8.2 Design science

With the exception of Paper IV, other artifacts have been designed and evaluated within a single application domain (e-commerce in Paper III and speed dating in Paper V). Being at an early stage of its development, the framework proposed in Paper IV currently has a limited number of algorithms and evaluation protocols, and should be interpreted as a proof-of-concept prototype.

3.8.3 Surveys

There are two main threats to validity when it comes to surveys. The first one has to do with the sample size. As mentioned earlier, the small sample size in Paper I is attributed to the fact that companies are expectedly reluctant to reveal the details of their algorithms. However, we have surpassed an acceptable minimum of 30 respondents recommended for surveys with a very low response rate [211].

The second limitation has to do with the representativeness of a sample. In Paper I, the sample is biased towards commercial platforms, whose number is almost twice that of open-source platforms. However, this distribution is simply a characteristic of the obtained responses, and not of our sampling strategy. In Paper II, on the other hand, we deliberately choose a sample that focuses on academic environments, and hence does not represent the entire population. This is a typical example of convenience sampling that is often practised in small-scale research [57] to obtain a sample in a fast and inexpensive way. We deem this choice acceptable for our *feasibility* study. In a full-fledged survey, however, a more representative sample would be desired. We partially compensate for this weakness by using probabilistic random sampling of target groups.

3.8.4 Case studies

A common criticism of case studies is that they typically do not generalize well [71]. The case study performed in Paper I focuses on "ordinary" small to medium size e-commerce platforms. Admittedly, the findings of the study are not likely to generalize to bigger players on the RS market (e.g. Amazon, Zalando, Criteo, etc.) who are very active in the RS research arena.

The significance of the case study conducted in Paper V may be impeded because of the small dataset size (4189 records). Although we use a large number of training folds in cross-validation (20), the accuracy of partner matchings remains low. This has not been an issue in the particular case as the focus of the paper is on the preferential fairness of recommendations rather than their accuracy. In practice, however, a fair but inaccurate recommender system would be of little use, and hence more training data would be normally required. Source triangulation has not been possible either due to the unavailability of suitable publicly accessible dating datasets. However, the theoretical contributions of the paper are applicable to other scenarios substantially beyond speed dating.

3.8.5 Literature review

In the literature study of Paper II, we have tried to identify as many RS-related ethical issues and incidents as was possible at the time of conducting this research. Being the first of its kind and intended as a discussion starter, this exploration is not to be taken as an exhaustive survey of all possible ethical considerations in RS. Besides, its main focus has been on the user-centric perspective, with only minimal account of multi-stakeholder recommendation ethics. See Milano et al. [190] for a wider coverage of that latter perspective.

We are also aware of the pitfalls associated with using the aforementioned "grey literature" sources. To mitigate the risk of running into false facts and misconceptions, we evaluate the credibility of each source by verifying the reputation of the respective author/resource before adding it to our reference list.

4 CONTRIBUTIONS

This section summarizes the contributions of the thesis. Each of the five formulated research questions is accompanied with sub-questions that focus on more specific aspects of the respective RQ. We now revisit each of them and elaborate on the findings of the related papers in response to these RQs.

4.1 Research question I

The first research question explores the gap between the research contributions of the RecSys communtiy and the realities of commercial recommender systems.

RQ I. How to assess the receptiveness of the e-commerce domain to the algorithmic innovations in the field of recommender systems?

This question is motivated and investigated in Paper I. As the first step to answering the question, it is important to understand and emphasize the specificities of the e-commerce domain in comparison to more "traditional" domains such as movies, for which many conventional RS have been designed. A vastly influential work is the one of Pradel et al. [230], who point out that "the abundance of academic papers on collaborative filtering should be contrasted with many critical aspects of the design of commercial recommender systems which remain mostly undocumented". In particular, it is noted that e-commerce datasets are characterized by very sparse purchase events and the general lack of ratings. Despite that many existing algorithms that were designed for rating prediction may still work on positive-only sales data after certain modifications, benchmarking their relative accuracy and speed on sales data would improve the understanding of their applicability to commercial settings. Having two proprietary e-commerce datasets at hand, we formulated the following sub-question:

RQ I (a). Which recommendation algorithms perform well on sales data?

We answer this question by running a simulation experiment in `MyMediaLite` [86] - a well-known RS library. Specifically, we evaluate representative techniques from three big algorithmic families - matrix factorization, neighborhood models (kNN), and association rules mining (ARM) - on two proprietary e-commerce datasets: fashion and books. We note that the ARM algorithm was not originally part of the library, and was implemented by us and contributed to `MyMediaLite` after the publication of Paper I. The results show that kNN and ARM are generally more accurate and faster when applied to sales data, as compared to two state-of-the-art matrix factorization methods, namely WRMF [113] and BPRMF [240]. Table 3 reports $F1@5$ measurements for different algorithms evaluated in random and chronological split modes (see Paper I for more detailed results).

Table 3: $F1@5$ scores of different algorithms run on two e-commerce datasets (Paper I)

		Collaborative filtering				Baselines	
Split	*Dataset*	*WRMF*	*BPRMF*	*kNN*	*ARM*	*Popular*	*Random*
Random	Fashion	0.1013	0.0676	0.1078	**0.1162**	0.0458	0.0000
Chronol.	Fashion	0.0188	0.0121	**0.0229**	0.0214	0.0113	0.0014
Random	Books	0.1322	0.0148	**0.1859**	0.1813	0.0149	0.0001
Chronol.	Books	0.0260	0.0130	0.0363	**0.0377**	0.0114	0.0000

Willing to investigate these somewhat surprising findings further, we formulated our next sub-RQ:

RQ I (b). Which recommendation algorithms are favored by industrial e-commerce platforms?

To answer the question, we perform a survey of existing commercial recommender systems on three criteria: (1) algorithms, (2) input data, and (3) properties. The results of the survey are summarized in Figure 2.

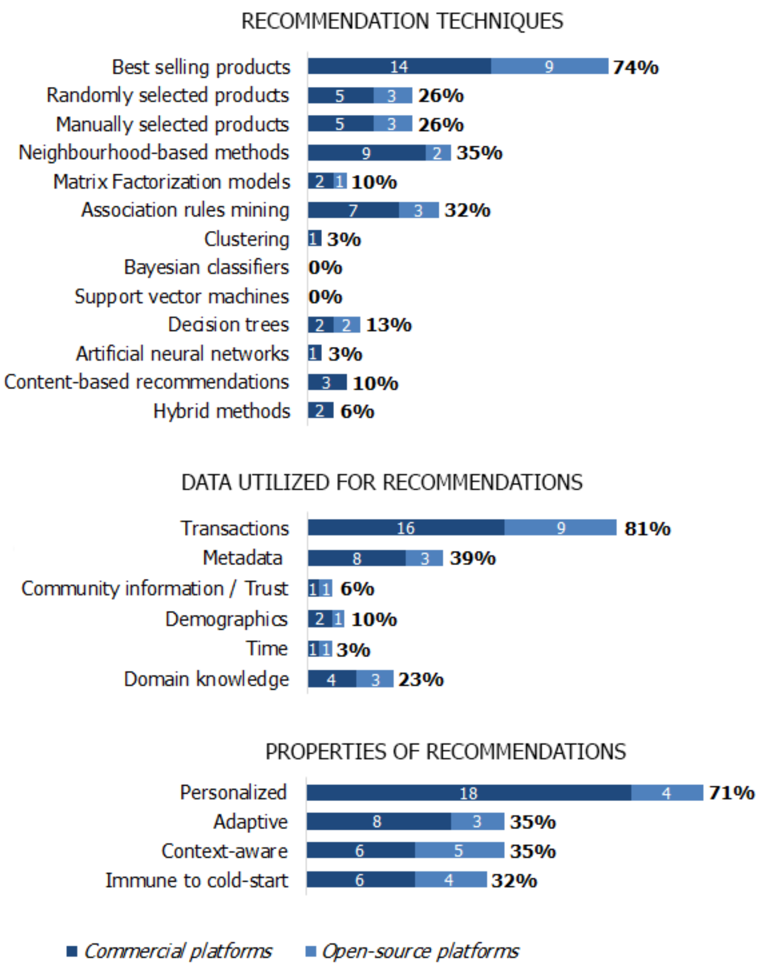


Figure 2: Characteristics of industrial recommendation engines (Paper I)

Triangulating the results of the survey with those of the experiment gives us a strong evidence for the practical attractiveness of neighborhood-based (kNN) and association rules mining algorithms for e-commerce data. In general, we observe that many platforms choose to display best sellers, manually selected products, and even random products as recommendations.

To summarize our findings in response to the main research question (RQ I), we observe that industrial e-commerce platforms favor rather simple algorithmic approaches, such as best-seller recommendations, kNN and association rules mining. This is reasonable in light of our experimental results, where more complex SVD-based methods do not seem to offer tangible advantages in accuracy and speed. Very similar results were reported in other studies on e-commerce recommendations [197, 230]. We also experiment with time-aware evaluation of RS, in line with some previous works [40, 160]. Specifically, we focus on the comparison between random and chronological splits, and establishing an optimal timespan for RS training. The results of the survey make it surprisingly evident that temporal information is not utilized by the majority of participating platforms[12].

As a side contribution, we devise a hyperparameter optimization technique based on *golden section search* as a faster alternative to the exhaustive grid search.

4.2 Research question II

The second RQ addresses the issues of ethics in RS, which emerge at various stages of the recommendation process.

RQ II. What are the ethical challenges that impact the design and use of recommender systems, and what are the possible solutions?

Prior to our research, the holistic coverage of RS-related ethical issues seems to have evaded the attention of the research community. The extensive literature study we perform in Paper II materializes in the list of ethical concerns that are mapped to each RS development stage. In a broad sense, ethical concerns are categorized into the ethics of data manipulation (e.g. privacy, anonymity, censorship issues), modeling (e.g. algorithmic biases, behavior manipulation, discrimination issues), and experimentation (e.g. (un)awareness, (un)equal treatment, informed consent issues). We also outline known solutions with respect to each of these issues. Our findings are summarized in the user-

[12]This observation only reflects the state of industry in 2014

centric ethical recommendation framework (Table 4), promoting the paradigm of "ethics awareness by design". The contribution of the framework is two-fold:

- to serve as a roadmap for RS practitioners who implement (ethics-aware) recommendation engines in real-world applications
- to raise the awareness around recommendation ethics in the RecSys community

Whereas it is arguably difficult and too early to speculate about the fulfilment of the first objective, the second goal has already been reached to a large extent through the works of Milano et al. [190, 191], who extended our research further to multi-stakeholder environments.

Aside from the theoretical exploration of recommendation ethics, we study the possibility of engaging the users themselves in addressing certain morally sensitive components of a recommender system. This is formulated as a secondary research question:

RQ II (a). How to aid morality in recommender systems through user engagement?

In response to this question, we propose a practical tool for RS users, which would give them the ability to manually adjust various morally-sensitive settings of a recommender system. We coin this feature an "ethical toolbox", where each filter corresponds to the specific stage of the articulated framework. For example, the initial *user profiling* stage allows users to restrict the creation and the maintenance of a user profile, whereas the last *testing stage* allows them to opt out of A/B testing experiments, and reset the recommendation engine to its "default" algorithm. The feasibility study we conduct in the frame of this research has a positive outcome, with 4 out of 5 proposed filters being preferred over alternative methods for handling a particular moral issue. Attention should be paid to the possible "control paradox" - a situation where users express their willingness to have control over their data, but do not make the effort of exploiting this control in practice [145].

Table 4: Ethical recommendation framework (Paper II)

Development stage	**User profiling** (input stage)	**Data publishing** (optional sharing stage)	**Algorithm design** (modeling stage)	**User interface design** (presentation stage)	**Online experiments** (optional testing stage)
Ethical concerns	Privacy breaches, lack of awareness / consent, fake profile injection	Privacy / security / anonymity breaches	Biases, discrimination, behavior manipulation	Algorithmic opacity, content censorship	Unequal treatment, side effects, lack of trust / awareness / consent
Known countermeasures	Informed consent, privacy preserving collaborative filtering, user identity verification	Privacy preserving data publishing	Algorithm audits, reverse engineering, discrimination-aware data mining	Explanations, ethical rule set generation, content analysis	Informed consent, possibility to opt out and delete data
User-adjustable controls (the toolbox)	"Do not track activity" tool *This setting restricts the creation and the maintenance of a user profile. Types of data can be manually defined and browsed items can be manually deleted (e.g. "manage history" tool on Amazon)*	"Do not share data" tool *This option allows local user profiling but restricts data sharing with third parties (even in the presence of anonymization). Types of data or categories of allowed recipients can be manually defined*	"Marketing bias" filter *This filter is used to remove any business-driven bias introduced by RS providers, and sets the recommendation engine to the "best match" mode (or other user-selected modes, such as "cheapest first")*	"Content censorship" filter *This tool can be used to set user-defined exclusion criteria for filtering out inappropriate items or categories. It also contains the option to turn the filter on and off (also with the possibility of scheduling)*	"Opt out of experiments" tool *This option can be used to reset the recommendation engine to its default algorithm, exclude the user from any future experiments, enable the opt-in option, and delete data from previous experiments*

This motivates further research towards the prototype implementation of the toolbox with subsequent user testing, which would allow us to draw more reliable conclusions about its usefulness and applicability.

4.3 Research question III

The insights gained from Paper I motivate the use of simple recommendation components (e.g. best-seller lists, co-purchased items, etc.) in an ensemble multi-arm bandit (MAB) algorithm. We are interested in simple components that can provide item-to-item recommendations on each new product page visit on an e-commerce site. The following research question is formulated:

RQ III. How to build a session-based context-aware bandit ensemble of elementary recommendation components with non-stationary rewards?

The solution we propose in Paper III models each recommendation component as a bandit arm, with some pre-defined policy orchestrating the arm selection process by balancing between exploration and exploitation. Two complications reflected in RQ III stem from the fact that the classical MAB formulation is not context-aware and that individual predictors (a.k.a. "weak learners") may have irregularities in their reward sequences. Our main contribution is the development of a bandit-driven ensemble based on Thompson Sampling, which has been adapted to meet the above requirements. In particular, we introduce the *dynamic partitioning* trick, which divides the output of each component predictor into disjoint itemsets with a (relatively) fixed probability of reward. In addition, the arm selection process is modified to introduce context-awareness by only disallowing use of components with empty responses (*sleeping bandits*). The results of Table 5 show how these modifications affect the accuracy of the proposed recommender system. This is our main contribution to RQ III.

Lastly, we adjust Thompson Sampling for the top-N ranking problem by employing a multiple-play mechanism, which performs one

Table 5: Standard vs. modified Thompson Sampling (paper III)

Agent	Recall	Precision	NDCG
	Books		
Thompson Sampling (SB+DP)	**0.0581** ↑83.3%	**0.0210** ↑94.4%	**0.0716** ↑94.6%
Thompson Sampling (SB)	0.0562 ↑77.3%	0.0202 ↑87.0%	0.0660 ↑79.3%
Thompson Sampling	0.0317	0.0108	0.0368
	Fashion		
Thompson Sampling (SB+DP)	0.1734 ↑21.3%	**0.0152** ↑13.4%	**0.0483** ↑19.9%
Thompson Sampling (SB)	**0.1760** ↑23.2%	0.0142 ↑6.0%	0.0427 ↑6.0%
Thompson Sampling	0.1429	0.0134	0.0403
	Yoochoose		
Thompson Sampling (SB+DP)	0.2973 ↑10.5%	**0.0419** ↑10.8%	**0.1391** ↑21.4%
Thompson Sampling (SB)	**0.2984** ↑10.9%	0.0409 ↑8.2%	0.1273 ↑11.1%
Thompson Sampling	0.2690	0.0378	0.1146

SB = Sleeping Bandit, DP = Dynamic Partitioning

arm pull for each recommendation slot in the list. With the above adjustments, the articulated bandit ensemble for e-commerce recommendations (BEER for short) can be driven by any off-the-shelf non-contextual bandit policy designed for stationary environments.

Our choice of Thompson Sampling is motivated by its proven practicality in modeling complex online problems [91], and its strong theoretical guarantees [135]. Still, it is worth comparing Thompson Sampling with other low-regret policies, primarily those based on Upper Confidence Bounds (UCB) [14, 159]. This is formulated as a separate sub-question:

RQ III (a). How does Thompson Sampling compare to other bandit policies for orchestrating the ensemble?

In the comparative study we perform in Paper III, Thompson Sampling outperforms six other popular MAB policies (MOSS, ε-greedy, and 4 variants of UCB) on all three datasets. It appears that Bayesian approaches in general are very effective in bandit problems with complex actions, since UCB-Bayes [135] comes second in accuracy in our experiments. We believe that these findings can be of interest to the research community, in particular as an additional evidence in support of Bayesian-oriented approaches advocated in [91].

Because Bayesian methods rely on priors, it is worth exploring whether the convergence of the ensemble can be improved by assigning more informative priors than the standard $Beta(1, 1)$ prior of Thompson Sampling. We formulate this as our second sub-question:

RQ III (b). How to prime the sampler with prior knowledge, and what effect does it have?

We utilize two sources of information for priming Thompson Sampler: (a) past user sessions, and (b) product catalog.

The first option allows us not only to obtain the initial beta posteriors for each component of the ensemble, but also to pre-train behavioral components (e.g. co-purchased items, co-clicked items, etc.) by running the sampler through recorded data. Our results show that using pre-trained behavioral components in the sampler leads to a consistent improvement in its accuracy.

We also demonstrate that it is possible to achieve an accuracy gain even when no historical data are available. In this case, we can take advantage of statistical properties of the product catalog to make informed guesses about the priors of attribute-based components (e.g. same author, same genre, etc.).

Our overall conclusion after experimenting with both approaches is that Thompson Sampling does not seem to enjoy particular benefits from pre-setting the Beta priors, meaning that it can handle cold-start gracefully out of the box. At the same time, our results encourage pre-training behavioral components with past transactional data, when such information is available.

Going beyond item-to-item recommendations, we formulate the final sub-RQ:

RQ III (c). How to personalize the ensemble with anonymous users, and what effect does it have?

We can achieve a weakly personalized behavior by modeling the relationship between anonymous browsing sessions. Specifically, we construct a kNN-based recommendation component with temporal decay capabilities. As Table 6 shows, the addition of this component to the ensemble leads to considerable accuracy gains.

Table 6: BEER(TS) with kNN personalization (paper III)

Agent	Recall	Precision	NDCG
	Books		
BEER[TS] with kNN	**0.0716** ↑31.1%	**0.0279** ↑30.4%	**0.0966** ↑64.0%
BEER[TS]	0.0546	0.0214	0.0589
	Fashion		
BEER[TS] with kNN	**0.1773** ↑1.0%	**0.0168** ↑8.4%	**0.0553** ↑9.5%
BEER[TS]	0.1756	0.0155	0.0505
	Yoochoose		
BEER[TS] with kNN	**0.3099** ↑3.9%	**0.0445** ↑5.0%	**0.1521** ↑6.9%
BEER[TS]	0.2983	0.0424	0.1423

To summarize the contributions to RQ III, we propose a bandit-based ensemble RS for streaming session data, driven by Thompson Sampling. The ensemble has a number of advantages for a prospective e-retailer:

- *Scalability*, because bandit arms are modeled as predictors (as opposed to single items), and hence scale well with the growing product catalog.
- *Extensibility*, because it is easy to switch between bandit policies, and add new component predictors on the fly.
- *Adaptability*, because the system dynamically reacts to non-responsive predictors and to the changes in reward probabilities.
- *Explainability*, because elementary predictors capture simple associations between items (e.g. co-purchases), and hence are easily interpretable.

4.4 Research question IV

Presently, there is a lack of evaluation tools and standards for stream- and session-based recommendations, which impedes their replicability and comparability [126]. To this end, we explore the possibility of adapting the `Scikit-Multiflow` stream learning framework for the session-based recommendation task. In particular, we focus on next-item recommendations.

RQ IV. How to adapt the `Scikit-Multiflow` streaming framework for rapid prototyping of session-based recommender systems?

To begin with, we cast the next-item prediction problem as a supervised online learning task. The goal is to predict the label y corresponding to the next item in a session, given the current context (e.g. session id, timestamp, etc.) encoded in the feature vector X. This enables us to use the *prequential evaluation* protocol provided by `Scikit-Multiflow`, with the following additions/modifications:

- A mechanism for evaluating a *list* of predictions against the true label, using *recall* and *mean reciprocal rank* metrics.
- Session support, which allows to separate the stream into (overlapping) user sessions.
- A sliding window of last k training instances, which can be used by recommendation models to get access to past session data, and to stay up-to-date by forgetting obsolete data.
- Support for different event types, which allows the evaluator to use different behaviors for different events, e.g. training on all types of events, but evaluating only on purchase events.

The above evaluation protocol allows to monitor the evolution of one or several recommenders in real time, which makes it easy to diagnose potential problems with learning (e.g. under cold start or concept drift), and also to better understand the specificities of a particular dataset (see Paper IV for some example visualizations). It also measures and reports training/testing runtimes for each algorithm. The averages of each evaluation metric are reported both on the global scale (i.e. for the entire simulation run), and within a sliding window of user-adjustable size.

One of the main advantages of using `Scikit-Multiflow` is the collection of stream learners that is provided by the library. Being designed for "ordinary" streams, these learners are not directly applicable to the recommendation problem. We therefore formulate the following sub-question:

RQ IV (a). How to utilize the underlying stream learners for the recommendation task?

One of the available stream learners is the Hoeffding Tree (HT), also known as the Very Fast Decision Tree (VFDT) [65]. In Paper IV, we develop a "wrapper" class for this algorithm, whose goal is to provide a recommendation interface for the underlying learner by handling its inputs and outputs. This wrapping procedure makes the learner suitable for online recommendation. The resulting HT wrapper algorithm was compared to several baseline streaming recommenders (including the BEER[TS] ensemble from Paper III), and was found to be consistently superior to them on three different datasets (see Paper IV for empirical results).

All the proposed modifications and algorithm implementations have been packed into the streaming recommendation framework titled `FlowRec`, which technically is an extension of `Scikit-Multiflow`. The framework has been released as an open-source Python library[13].

4.5 Research question V

In continuation of exploring the ethical dimension of RS, we conduct a case study of fair matchmaking in speed dating. This study was partially inspired by the CoffeeMeetsBagel incident [210], when the part-

[13] **`https://git.io/flowrec`**

ner recommendations provided by the dating platform were perceived by many users as racially biased. The following research question is formulated:

RQ V. How to define and reason about the fairness of match-making in the context of speed dating, with respect to the expressed preferences for sensitive attributes of users?

In response to RQ V, we derive a definition of *preferential fairness*, and put it in the context of existing fairness formalizations in machine learning. In particular, we show that preferential fairness is a special case of *calibrated* fairness, where the user's expressed preference for the partner's sensitive attribute and its distribution in the candidate pool set the merit for choosing the right candidates for recommendation. This merit can be expressed numerically as the optimal (in terms of fairness) mean value of the sensitive attribute at a given size of the recommendation list. This is the first step to quantifying preferential fairness, which is the subject of the first sub-RQ:

RQ V (a). How to measure *preferential fairness*?

The calculation of preferential fairness is based on the distance between the current mean and the optimal mean of a recommendation list for user u. Let us denote this distance (a.k.a. offset) as δ_u. Likewise, let δ_{min} and δ_{max} define the minimum and the maximum possible offsets from the optimal mean for the current size of the list. We can now define the preferential fairness on a [0,1] scale as follows:

$$\phi_u \stackrel{\text{def}}{=} 1 - \frac{\delta_u - \delta_{min}}{\delta_{max} - \delta_{min}} \tag{16}$$

Higher values of ϕ_u suggest greater fairness, which can also be conveniently expressed as a percentage score. In principle, it can be useful in any other domain where explicit user preferences form the basis for establishing the fairness of recommendations. Refer to Paper V for a more detailed derivation of ϕ_u.

Apparently, the preferential fairness of recommendations may be at odds with their accuracy. This is the subject of the second sub-RQ:

RQ V (b). How to resolve the fairness-accuracy trade-off?

The idea is to re-rank the generated recommendation list in a post-processing step such that the desired level of fairness is achieved with minimal accuracy compromises. We propose formulating the fairness-accuracy trade-off problem as a 0-1 Knapsack problem, which can be solved with dynamic programming or other heuristics, such as Tabu search. The former method guarantees an optimal solution in pseudo-polynomial time for binary attributes. The latter method is based on a hill climbing heuristic, which is designed to find a good approximation fast, and potentially scales better for larger candidate pools. We demonstrate how to use both methods in the binary case of racially biased predictions, as well as for the non-binary case of religiously biased predictions. The proposed approach allows a system provider to set a tolerance threshold for how much accuracy can be sacrificed for increased fairness, or vice versa.

Our experimental results show that the preferential fairness of off-the-shelf matchmaking algorithms does not exceed 55% for race, and 59% for religion. After the re-ranking procedure, the largest accuracy drop suffered due to enforcing 100% fairness is 4.4% for the binary attribute (race), and 9.7% for the non-binary attribute (religion). The proposed methodology is justifiable on the grounds of modest accuracy compromises, and of the legal and ethical ramifications of racially and religiously biased predictions. It also has a theoretical value in the proposed variants of the Knapsack problem and Tabu search. The details are provided in Paper V.

5 CONCLUSIONS

The present thesis touches upon some pragmatic aspects of recommender systems related to their algorithmic and ethical dimensions. This pragmatism is grounded in the increasing demand to understand the applicability of existing recommendation algorithms and evaluation protocols to the realities of the industrial sector, as well as the ethical impact of this technology.

Our initial insight into industrial recommendation engines emphasize the practical advantage and the wide spread of relatively simple algorithmic approaches in real-world applications, despite the academic focus on more elaborate matrix factorization-based models. This resonates well with the Netflix' business decision not to implement the prize-winning algorithm due to unjustified engineering efforts [9]. Five years later, when deep learning models are dominating the field of RS, most of them can still be outperformed by substantially simpler algorithms such as kNN [54]. This also holds for session-based recommender systems [120, 173, 174], which have become increasingly relevant in recent years for modeling short-term user interests in anonymous sessions.

Taking this as a starting point, we devise a bandit-based algorithm for combining a set of simple recommenders in an ensemble learning scheme, which can generate session-based recommendations in streaming mode [32]. Stream- and session-based RS have practical value because they reflect the sequential nature of user interactions on a website and retain the actual behavioral structure of historical data. The proposed ensemble has been designed for purchase prediction in e-commerce, and has proven especially effective with Thompson Sampling at its core. Our second algorithmic contribution is the Hoeffding Tree-based recommender system for next-item prediction, which belongs to the class of (online) supervised learning models [218]. The proposed method operates in an ensemble-like manner when making predictions, and has shown consistent superior performance compared to the considered baselines. Both recommenders are characterized by their relative simplicity, interpretability, and adaptability to evolving data streams from the initial state of cold-start. We provide open-source implementations of these recommenders in the proposed simulation framework that we coin `FlowRec`[14]. The framework has been developed on top of `Scikit-Multiflow` — a machine learning library for streaming data. `FlowRec`'s mission is to provide a tool for rapid prototyping of session-based recommendation algorithms in streaming mode, by offering standardized evaluation protocols and recommender implementations. The "rapidness" of prototyping is also warranted by the immediate access to numerous stream learners readily available via the `Scikit-Multiflow`'s code base, which can be utilized for recommendation purposes with reasonably little effort, as we demonstrate in [218].

The ethical side of recommender systems has been largely overlooked by academic researchers because of the fact that RS prototypes rarely leave the lab environment. In real-world applications, the moral impact of recommendations deserves serious consideration, since they directly influence users' choices, and can be susceptible to social effects such as bias reinforcement, discrimination, misinformation spread, filter bubbles, and so on. Our research is the first attempt to formally define the concept of *recommendation ethics*, and to systematize various scarcely reported ethical issues and the respective countermeasures into a coherent framework. This analysis has been done by following the typical recommendation pipeline and identifying/exemplifying potential ethical concerns that emerge at various RS development stages. Addressing these concerns implies resolving many challenging trade-offs, such as user privacy vs. personalization, data anonymization vs. data utility, informed consent vs. experimentation bias, and algorithmic transparency vs. trade secrets. The results of this analysis can serve as a roadmap for building recommender systems that are "ethics-aware by design". In addition, we have explored the idea of engaging users in adjusting a recommender system according to their own moral standards, which has been well received by the participants of the per-

[14]Note that the `FlowRec`'s implementation of BEER[TS] ensemble is slightly different from the one described in the original paper [32]; see [218] for details.

formed feasibility study. However, the idea is to be taken as a half-measure due to the extra burden that it puts on users. We therefore advocate the emergence of recommender systems with autonomous moral agency that is grounded in existing theories and notions of social justice and fairness. To this end, many fairness-aware algorithms and metrics have recently appeared on the RS horizon [36, 168, 265, 310, 314, 327, 328]. We continue in this direction by focusing on fair matchmaking in dating services [219]. We introduce the notion of *preferential fairness*, which is achieved by mapping the user's expressed preference for a sensitive attribute to the distribution of that attribute in the candidate pool. We then propose two re-ranking methods for solving the recommendation problem under fairness constraints. The proposed methodology has been successful in minimizing the fairness-accuracy trade-off in matchmaking with racial and religious bias.

The recent self-reflectory insights of the RecSys community [54] show worrying signs of the "phantom progress", where the primary choice of values (e.g. the hunt for accuracy) and algorithms (e.g. deep learning) appear to be misplaced and often distant from the real-world application needs. This is in line with the tone of the present thesis, which advocates academic research endeavors that are driven by the practical and ethical appropriateness of recommendations.

PART II

INCLUDED PUBLICATIONS

6 PAPER I

Paraschakis D., Nilsson B.J., and Holländer J.

"Comparative Evaluation of Top-*N* Recommenders in E-commerce: an Industrial Perspective"

Abstract We experiment on two real e-commerce datasets and survey more than 30 popular e-commerce platforms to reveal what methods work best for product recommendations in industrial settings. Despite recent academic advances in the field, we observe that simple methods such as best-seller lists dominate deployed recommendation engines in e-commerce. We find our empirical findings to be well-aligned with those of the survey, where in both cases simple personalized recommenders achieve higher ranking than more advanced techniques. We also compare the traditional *random* evaluation protocol to our proposed *chronological* sampling method, which can be used for determining the optimal time-span of the training history for optimizing the performance of algorithms. This performance is also affected by a proper hyperparameter tuning, for which we propose *golden section search* as a fast alternative to other optimization techniques.

Published in Proceedings of the 14th International Conference on Machine Learning and Applications (ICMLA), IEEE, 2015

6.1 Introduction

An intelligent recommender system (RS) is a powerful selling tool of any contemporary e-commerce platform. According to Forrester[15], product recommendations account for up to 30% of e-commerce sites' revenues. That was already apparent in 2006, when Amazon[16] claimed that 35% of its sales were triggered by recommendations [179]. The same year, Netflix[17] announced its famous million-dollar prize for improving its recommendation engine. This contest has spurred active research within the field, yielding a variety of recommendation algorithms over the past decade. Interestingly, the prize-winning algorithm was never put to real use: Netflix concluded that the measured accuracy gains "did not seem to justify the engineering effort needed to bring them into a production environment" [9]. This fact motivated us to explore how receptive *e-commerce* platforms are to academic advances in the field of recommender systems. The connection between research contributions and industrial solutions has rarely been in focus of RS literature. As Pradel et al. [230] noted, "case-studies are necessary to better understand the specificities of purchase datasets and the factors that impact recommender systems for retailers". Our work is such a case study.

Originating from the Netflix problem, the vast majority of proposed recommenders have been designed for rating prediction. In e-commerce, however, rating products is not commonly practised even when this functionality is supported by a shopping platform. The problem thus boils down to predicting a purchase rather than a rating score. Purchase prediction based solely on transactional history is the common approach known as *collaborative filtering* (CF). In the absence of explicit ratings, transactional datasets become binary and are referred to as *implicit feedback* datasets. When it comes to missing data, implicit feedback is ambiguous: if a customer has never bought a particular product, it is not always a sign of negative preference (e.g. the product might be out of stock). That is why algorithms designed for implicit

[15] **http://www.forrester.com**
[16] **http://www.amazon.com**
[17] **http://www.netflix.com**

feedback often model heuristics of relationships between observed and unobserved events in a dataset, as we shall see in Section 6.9.1. This direction is less researched than movie ratings, partly due to the lack of publicly available e-commerce datasets. In our experience, retailers are reluctant to release their sensitive purchase data. Another factor that differentiates these datasets from movie ratings is the extreme sparsity of data: a typical retail store only sells a fraction of its product catalog.

Evaluation of recommender systems in industrial context deserves special attention. Whereas the main academic interest appears to be the accuracy of predictions, industrial recommenders are meant to be optimized for revenues. As Aiolli [5] pointed out, the choice of an accurate predictor is not all that is required to build a good recommender. Other evaluation criteria that play an important role in e-commerce are coverage of product catalog, utility and serendipity of recommendations, adaptivity and scalability of an algorithm, etc. [95]. Since our work aims at a comparison between algorithms and not a rigorous testing of a particular recommender, we only consider accuracy and speed as our evaluation criteria. The most reliable way of testing a deployed RS is to perform an online evaluation (e.g. A/B testing), which may however be too costly, time-consuming and therefore often infeasible in practice. That is why customer behaviour is usually simulated offline from the recorded purchase data. The choice of a proper user modeling strategy is a crucial factor, which must provide a realistic estimate of the real performance of an algorithm. Numerous approaches for such modeling have been discussed in [95]. In this paper we propose one such technique for offline evaluation.

The contributions of this work are the following:

1. We conduct a survey of deployed recommendation engines in the e-commerce domain. This is done to measure the extent to which recent academic achievements in the field of recommender systems are adopted by real-world applications. To the best of our knowledge, this is the first study of this kind.
2. We find that the performance of simpler but faster algorithms such as association rules can often be superior to more elaborate techniques such as matrix factorization (MF). This confirms the results of other recent works on purchase data, such as [230].
3. Our experiments show that training on all available data is inferior to training on recent purchases for most algorithms. To optimize recommendation accuracy, we propose the expanding time window evaluation method for finding the optimal training time-span.
4. We adapt the Golden Section Search (GSS) algorithm for multi-dimensional hyperparameter optimization.

The paper is organized as follows. Section 6.3 presents our methodology, where we describe our datasets, evaluation protocol and metrics. We then present the GSS algorithm and the survey. In Section 6.9, we give a brief presentation of the recommendation techniques used in our study. In Section 6.10, we present and analyze our experimental results and survey responses. Finally, Section 6.11 concludes the paper.

6.2 Related work

A few case studies, closely related to ours, performed comparative evaluations of different families of recommendation algorithms on purchase datasets. A study by Huang et al. [115] points to the need for better understanding of relative strengths and weaknesses of different types of algorithms in e-commerce applications. Thus, they performed a comparison of six different algorithms and found that *spreading activation* and *link analysis* produced the best results. Just as in our study, they experimented on a fashion and a book store. A more recent case study by Pradel et al. [230] experimentally evaluated various CF algorithms on a dataset of a French home improvement and building supplies chain. The simple *bigram rules* recommender was found to yield the best overall accuracy. One important conclusion of this work is that the relative performances of algorithms depend on the *setting* to which they are applied.

In relation to this, both studies evaluate recommenders in two settings: complete and reduced transaction history. In [115], the reduced training set contains a fraction of randomly sampled events, whereas in [230] it contains the event history of the last two weeks of the training data. In our paper, we adopt a more flexible setting where we evaluate algorithms in various history lengths (see Section 6.5).

6.3 Methodology

In our study we conduct offline experiments on two real e-commerce datasets: a fashion store (denoted as D1) and a book store (denoted as D2). All experimentation is done on an Intel Xeon 2.53 Ghz PC with 12 Gb RAM. Our offline experiments aim to compare the performance of algorithms from the three families: MF-based CF, memory-based CF, and data mining (detailed in section 6.9). Two methods that recommend most popular and random products are used as baselines. The representative algorithms from each of the aforementioned families were selected from the MyMediaLite library[18] [86] to aid reproducibility. Semantic- and context-aware methods have not been considered due to the nature of the input data, which consists solely of purchase events. All techniques are evaluated for their predictive power and speed. Our experiments are complemented with an online survey, which assesses recommendation engines of existing e-commerce platforms. Survey responses are analyzed in Section 6.10.3.

6.4 Datasets

The summary of the two datasets is given in Table 7.

[18] with the exception of the ARM recommender and the LLR-based similarity, which were implemented separately

Table 7: Summary for datasets D1 and D2

	D1	**D2**
Domain	Fashion	Books
Timespan	9 months 26 days	3 months 10 days
Customers	26,091	505,136
Products	4706	210,455
Events	78,449	1,623,576
Sparsity	99.936%	99.998%

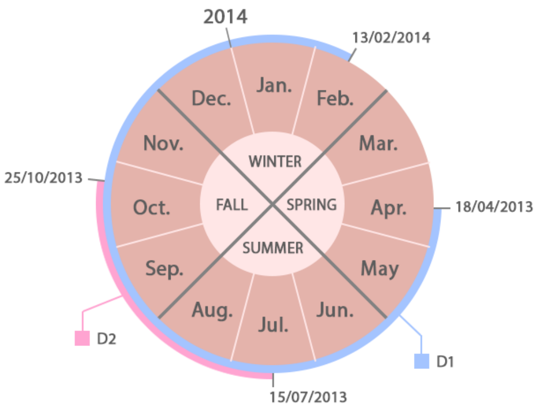


Figure 3: Timespan of datasets

Both datasets are of implicit feedback and contain a history of purchases over a period of time. Each dataset is represented as a binary purchase matrix $P^{[m \times n]}$, whose entry $P_{ui} \in \{0, 1\}$ indicates whether customer $u \in U^{[m]}$ bought product $i \in I^{[n]}$ at least once. The two datasets differ in size, sparsity, type of merchandise, and timespan (the timespan is illustrated in Figure 3).

The distribution of purchases per customer is given in Figure 4, which depicts the characteristic sparsity of purchase data. For our experiments, we pre-filtered the datasets to contain customers with at least 2 purchases, which is the bare minimum that enables customer profile splitting (see section 6.5), while maintaining high sparsity levels.

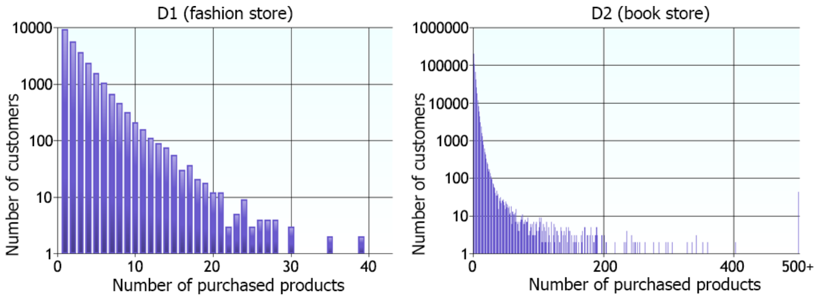


Figure 4: Distribution of purchases per customers

6.5 Experimental protocol

The common strategy for offline experiments is to split the dataset into training and test sets, where the latter simulates future purchases and contains a fraction of events withheld from the original dataset. The remaining events are kept in the training set, which is fed to a recommendation engine that outputs a ranked list of unobserved products based on some underlying algorithm. The top-N products are then taken from the list as final recommendations. The choice of the split ratio between training and test sets is arbitrary and usually ranges from 80/20 to 50/50 percent.

In our experiments, we follow two splitting strategies:

1. **Traditional approach (non-chronological split)**. As it is often practiced in RS literature [139, 240, 249, 255], we start by discarding "cold-start" customers with profiles below a constant threshold. In our case, we only keep those with at least 10 purchases. We then divide all customers into 5 folds for cross-validation. In each test fold, we randomly withhold half of events[19] from customer profiles to build the test set. The other half of events together with the remaining folds is used for training the models.

2. **Proposed approach (chronological split)**. The availability of timestamps allows us to attempt a more realistic setup, where we train on past events to predict future events. Furthermore, we deliberately

[19]if the number of events is odd, the training set will be larger than the test set by 1 event

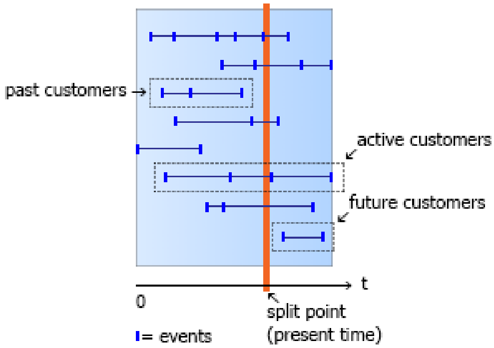


Figure 5: Representation of a temporal dataset. Horizontal lines represent customer profiles on a timeline, where purchases are marked with blue bars.

keep all customers in place from the initial pre-filtering step, when only those with at least 2 purchases were kept. This is the shortest user profile length that can be divided in two parts (training and testing). This choice was motivated by our intention to simulate a real cold-start environment. A temporal dataset is illustrated in Figure 5.

In this setting, we set a split point on the dataset's timeline (indicated with the red vertical line), which acts as our "present". Past events located to the left of the split point are used for training, whereas future events located on the right side are used for testing. Customers whose entire profiles lie in the "future" are discarded, whereas "past customers" only appear in the training set. Keeping the split point (and hence the test set) fixed, we train the models using the expanding time window, starting from the most recent history and then going back in time at a fixed rate up to full history. This approach allows us to determine an optimal training interval from our measurements.

In both cases, the task is to recommend products matching those of the test set. The motive for attempting both approaches was to determine whether the widely used random, non-chronological splits can reliably reproduce the same ranking of algorithms as the more realistic temporal, cold-start splits.

6.6 Evaluation metrics

Top-N recommendation lists are typically evaluated in terms of their precision and recall. Precision measures the percentage of recommended products that are relevant,[20] whereas recall measures the percentage of relevant products that are recommended.
Given a fixed size N of recommendation list, we calculate the harmonic mean of precision and recall, known as $F1\text{-}score$:

$$F1@N = \frac{1}{|\mathcal{U}|} \sum_{u=1}^{|\mathcal{U}|} \frac{2 \cdot precision_u@N \cdot recall_u@N}{precision_u@N + recall_u@N} \tag{17}$$

where $\mathcal{U}$ is the set of "active customers" (see Figure 5).

In our experiments we choose $N = 5$. Notably, evaluation at fixed N suffers from bias introduced by unevenly distributed number of events in the test set across customers. When this number is larger than N, even a perfect recommender would never reach the maximum accuracy score. Therefore, metrics with fixed N do not average well and can only be used to estimate the relative performance between algorithms. To tackle this problem, we also measure precision at the level of recall, known as $R\text{-}precision$:

$$R\text{-}precision = \frac{1}{|\mathcal{U}|} \sum_{u=1}^{|\mathcal{U}|} precision_u@|T_u| \tag{18}$$

where T_u is the test set of customer u.

Thus, $R\text{-}precision$ adjusts for the size of the customer u's test set and hence ensures that evaluation has an attainable maximum value [249]. Thus, the aforementioned bias is eliminated and averaging across customers makes more sense [177]. As opposed to $F1\text{-}score$, $R\text{-}precision$ allows to estimate the absolute quality of a recommendation algorithm [249].

The disadvantage of the above evaluation metrics is that they do not capture the utility of recommendations, making them all look equally

[20] being *relevant* means being a member of the test set

interesting to the customer. However, usability studies show that customers are used to scan recommendation lists from top to bottom, often observing only a few products on the top of the list [95]. Therefore, we use a special ranking measure to account for a positional discount down the list, known as Mean Average Precision (*MAP*):

$$MAP = \frac{1}{|\mathcal{U}|}\sum_{u=1}^{|\mathcal{U}|}\frac{1}{|T_u|}\sum_{k=1}^{|T_u|} precision_u@k \tag{19}$$

where $precision_u@k = 0$ if $product\_at\_position\_k \notin T_u$

Thus, *MAP* is the arithmetic mean of precision values across recall levels, averaged over all customers. *MAP* is a very popular metric in information retrieval and is favored for its discrimination and stability [177].

The above evaluation metrics are measured within the range [0,1], where higher values are better.

6.7 Hyperparameter optimization via Golden Section Search

Most recommendation algorithms are affected by their hyperparameters that need to be properly "tuned" when applied to a new dataset. This is a problem of optimizing a cost function over a (multi-dimensional) configuration space. Evaluating a cost function is often computationally expensive, rendering common approaches such as manual and grid search inefficient. We address this issue by adapting a Golden Section Search (GSS) to our multi-dimensional hyperparameter optimization, under the assumption of unimodality of the objective function. In one dimension, the problem can be formulated as finding a point x in the configuration space $[a, b]$ such that the objective function $f(x)$ is maximized:

$$x^* = \operatorname*{arg\,max}_{a \leqslant x \leqslant b} f(x) \tag{20}$$

The GSS algorithm finds x^* through iterative shrinkage of the configuration space by sampling points between a and b using the inverse golden ratio $\varphi \approx 0.618$, until the resulting interval becomes smaller than the pre-set tolerance ϵ:

Algorithm 2 Golden Section Search

Input: List $\mathcal{P}(param_1\langle a_1, b_1, \epsilon_1\rangle, \ldots, param_n\langle a_n, b_n, \epsilon_n\rangle)$

Output: Optimal values $x_1^*, ..., x_n^*$ of parameters in $\mathcal{P}$

function MULTI-GSS($\mathcal{P}$)
    while $|\mathcal{P}| > 0$ **do** $\triangleright$ unoptimized parameters exist
        for $i \leftarrow 1,|\mathcal{P}|$ **do**
            $x_1 \leftarrow b_i - \varphi(b_i - a_i)$
            $x_2 \leftarrow a_i + \varphi(b_i - a_i)$
            if $f(x_1) < f(x_2)$ **then** $\triangleright$ x_i^* must be in $[x_1, b]$
                $a_i \leftarrow x_1$
            else $\triangleright$ x_i^* must be in $[a, x_2]$
                $b_i \leftarrow x_2$
            if $b_i - a_i \leqslant \epsilon_i$ **then** $\triangleright$ $param_i$ is optimized
                $x_i^* \leftarrow (b_i - a_i)/2$
                remove $param_i$ from $\mathcal{P}$
                return x_i^*

Given a set $\mathcal{P}$ of hyperparameters, the algorithm iterates over them and the function $f(x)$ evaluates a recommender in the current hyperparameter setting. Based on the given values of a, b, and ϵ of each hyperparameter, the required number of iterations can be pre-computed as follows:

$$\#iter = \frac{\sum\limits_{i \in \mathcal{P}} \log \frac{\epsilon_i}{b_i - a_i}}{\log \varphi} \tag{21}$$

This allows for adjusting the value of ϵ prior to running GSS in accordance with the available computational resources. GSS has fast convergence because it never samples past a suboptimal point, but does not guarantee finding a global maximum when the objective function is multimodal. This leads to situations when the last iteration does not necessarily produce the highest function value. To tackle this problem, we save the values of hyperparameters every time when the evaluated function peaks at its current best value.

For tuning hyperparameters in our chronological setup, we used validation sets dating from 17/07/2013 to 31/07/2013 for D1 and from

29/07/2013 to 05/08/2013 for D2. We trained all algorithms on events prior to the validation events of each dataset. For the non-chronological case we validated algorithms on customers with 9 events, where random 65%-35% user profile splitting was done. This ensures that our validation sets are always disjoint from the test sets, where only customers with at least 10 events are considered.

6.8 Survey of deployed recommender systems

The survey was aimed at commercial and open-source e-commerce software vendors, who were asked to describe their recommendation engines as accurately as possible. This was done via an anonymous online questionnaire, which assessed the deployed RS on three criteria: a) properties of recommendations, b) data utilized for recommendations, and c) recommendation techniques. The selection of these techniques is based on the classification by Amatriain et al. [10], with the addition of extra (non-personalised) techniques, such as random, manual, or best selling product recommendations. Each question had a set of pre-defined answers with a possibility to select multiple options and provide a free-form answer if so desired. The questionnaire is available online at **`http://goo.gl/forms/zFEfLLlFHO`**.

6.9 Recommendation algorithms

In this section we give a brief theoretical foundation for various algorithms that were used in our experiment.

6.9.1 MF-based methods

MF-based models decompose the original purchase matrix P into lower-rank factor matrices $W \in \mathbb{R}^{[m \times d]}$ and $H \in \mathbb{R}^{[d \times n]} (d \ll min(m, n))$, which capture latent relationships between customers and products. The interest of customer *u* in product *i* is estimated by the dot product of the corresponding factor vectors $\hat{P}_{ui} = W_{u:} H_{i:}^T$. The major challenge is to compute the mapping of customers and products to a joint latent factor space.

Most prevalent factorizations encountered in RS literature are based on singular value decomposition (SVD), where the mapping is performed iteratively either through the alternating least squares (ALS) or the stochastic gradient descent (SGD) optimization process. In our study, we examine several SVD-based algorithms which have been specifically designed for implicit feedback datasets.

WRMF The Weighted Regularized Matrix Factorization model [113, 215] is a state-of-the-art algorithm based on SVD, which is formulated as a least squares problem with two core components: a regularization term to prevent overfitting, and a confidence value associated with each event. The intuition behind this measure is that the confidence of observed events being positive is higher than the confidence of unobserved events being negative. One way to enforce this is to assign higher weights to positive entries in P, although other weighting schemes have also been discussed by Pan et al. [215]. The WRMF optimization criterion is given as:

$$min \sum_{ui} C_{ui}((P_{ui} - W_{u:}H_{i:}^T)^2 + \lambda(||W_{u:}||_F^2 + ||H_{i:}||_F^2)) \tag{22}$$

where C_{ui} is a confidence of event P_{ui}, λ is a regularization parameter, and $|| \cdot ||_F$ is a Frobenius norm of a matrix.

The factor matrices are first initialized with normally distributed random numbers. The optimization process then alternates between updating W and H as follows: [215]:

$$W_{u:} := P_{u:}C_{u:}H(H^T C_{u:}H + \lambda I)^{-1} \tag{23}$$

$$H_{i:} := P_{:i}^T C_{:i}W(W^T C_{:i}W + \lambda I)^{-1} \tag{24}$$

The advantage of ALS is that it can be easily parallelized and its running time is linear in the size of the input [113].

BPRMF The BPRMF [240] model is formulated as a ranking problem and is based on Bayesian Personalized Ranking optimization criterion (BPR-OPT). Unlike WRMF which scores single products in P, BPRMF optimizes for correct ranking of product pairs for each customer, i.e. the triple $\langle u, i, j \rangle$. The basic idea of BPR is that if customer u consumed product i but did not consume product j, then $P_{uj} < P_{ui}$ ranking-wise. The generic BPR-OPT criterion is given as [240]:

$$\textit{BPR-OPT} = \sum_{uij} \ln \sigma(\hat{x}_{uij}) - \lambda_\theta ||\theta||^2 \tag{25}$$

where $\sigma(x)$ is a logistic sigmoid $1/(1+e^{-x})$, $\hat{x}_{uij}$ is an arbitrary function that specifies the relationships in $\langle u, i, j \rangle$, θ is a parameter vector of an arbitrary model, and λ_θ is a model-specific regularization parameter.

In BPRMF, the estimation of $\hat{x}_{uij}$ is performed through matrix factorization. But since the MF model can only predict single events, the estimator is decomposed into single prediction tasks: $\hat{x}_{ui} - \hat{x}_{uj}$. The optimization is performed through SGD with bootstrap sampling of training triples using the following update rule [240]:

$$\theta := \theta + \alpha\left(\frac{e^{-\hat{x}_{uij}}}{1 + e^{-\hat{x}_{uij}}} \frac{\partial}{\partial \theta} \hat{x}_{uij} + \lambda_\theta \theta\right) \tag{26}$$

where α is the learning rate.

6.9.2 Data mining (association rules)

Association rules mining (ARM) has been a popular instrument for market basket analysis (e.g. in supermarkets), where associations between products are established from customer purchase patterns. This technique is therefore a natural choice for recommender systems and has demonstrated good performance on real-world e-commerce datasets, as shown by Pradel et al. [230]. In its simplest form, an association rule $i \Rightarrow j$ implies that whenever a customer buys product i, he or she is likely to buy product j. The strength of a rule is measured on the basis of its confidence and support, which are calculated as follows:

$$con(i \Rightarrow j) = \frac{P_{:i}^T P_{:j}}{\sum\limits_{u \in U} P_{ui}} \qquad\qquad sup(i \Rightarrow j) = \frac{P_{:i}^T P_{:j}}{\sum\limits_{u \in U} \sum\limits_{i \in I} P_{ui}}$$

In other words, confidence is the probability of buying product j given that product i was bought, while support is the fraction of events where products i and j were bought together.

The training phase involves computation of all the available rules, whose confidence values are stored in a sparse matrix of size $n \times n$. To produce recommendations for customer u, a set of supported rules is first identified from a customer profile. Then, unobserved products are typically ranked by either the maximum confidence [230, 255] or the sum of confidences [139] of corresponding rules. In our case, the linear combination of confidence and support was found to yield better results:

$$rank(j) = \sum_{i}^{|I_u|} con(i \Rightarrow j) \cdot sup(i \Rightarrow j) \tag{27}$$

where I_u is the profile of customer u (e.g. the set of observed products), and j is a candidate product for recommendation.

6.9.3 Memory-based CF (K-nearest neighbors)

Memory-based CF relies on the notion of similarity between customers or products. Thus, the key training step is the formation of a neighborhood, which becomes a source of recommendations. For a given customer, recommendations can be computed either from the neighborhood of similar customers (user-based kNN), or from the neighborhood of products similar to those he or she has bought (item-based kNN). Although the latter approach has been favored in RS literature, we found user-based kNN to be more effective on both our datasets, in line with the results of [5, 300].

The performance of a kNN recommender is much dependent on the similarity measure used to calculate the neighborhood. Cosine and Jaccard similarity are commonly used for this task. In our tests, the Log-Likelihood ratio proposed by Dunning [66] and adapted for CF by Owen et al. [214] produced superior results. In short, this metric estimates the probability that the overlap of events in customer profiles is not due to chance [214]. Let I_u and I_v denote two customer profiles. The Log-Likelihood ratio (LLR) is then defined as:

$$LLR = f(I_u, I_v) \cdot (H[I_u, I_v] - H[I_u] - H[I_v]) \tag{28}$$

where $f(I_u, I_v)$ is a count-dependent scaling factor, $H[I_u, I_v]$, $H[I_u]$

and $H[I_v]$ are the entropy of the joint and the marginal probabilities, respectively (refer to [66, 67] for details).

During training, an LLR-based neighborhood is calculated for each customer. The testing phase ranks unobserved products based on the accumulated LLR scores of each neighbor who consumed the products in question.

6.10 Results

In this section, we present the results from our experiments on datasets D1 and D2. All algorithms were tuned using GSS and then evaluated on the three metrics described in Section 6.6. After reporting the experimental results, we also present a summary of the survey responses.

6.10.1 Experiment 1. Non-chronological split

We first present the results from the "traditional" approach, where we have:

- test customers with at least 10 purchases
- random sampling of events (50/50 profile splitting)
- 5-fold cross-validation

The performance of algorithms in this setup is given in Tables 8 and 9. It can be seen that User$k$NN and ARM outperform MF-based methods on all metrics, which is especially prominent in case of D2, where BPRMF has performance comparable to that of the most popular books recommender. This finding conforms to those reported in [5, 230].

The presented runtimes of algorithms were calculated as the sum of training and testing time. In Tables 10 and 11 the average running times (across training sets) are reported. The results are given in the format *hh:mm:ss*. Aside from the implementation details, the running time of algorithms depends on their hyperparameter values, such as K for User$k$NN or *iterations* / *factors* for MF-based methods. We can see that WRMF is consistently slower than other methods, despite having been trained with fewer iterations than BPRMF in all cases. We attribute it to the fact that ALS involves a least squares solution, which is more expensive than an SGD iteration. The noticeable boost in WRMF speed in Table 11 is merely because the algorithm was trained with fewer factors than in random mode (determined by the GSS).

Table 8: Scores for the random split on D1

Recommender	*MAP*	*F1@5*	*R-prec*	*Running Time*
WRMF	0.0927	0.1013	0.1036	00:05:21
BPRMF	0.0650	0.0676	0.0698	00:00:32
User$k$NN	0.1000	0.1078	0.1073	00:00:05
ARM	**0.1100**	**0.1162**	**0.1155**	00:00:01
Most Popular	0.0523	0.0458	0.0468	00:00:00
Random	0.0028	0.0000	0.0010	00:00:00

Table 9: Scores for the random split on D2

Recommender	*MAP*	*F1@5*	*R-prec*	*Running Time*
WRMF	0.1293	0.1322	0.1363	07:10:10
BPRMF	0.0188	0.0148	0.0187	00:12:46
User$k$NN	**0.1796**	**0.1859**	**0.1854**	00:56:21
ARM	0.1774	0.1813	0.1825	00:16:24
Most Popular	0.0139	0.0149	0.0156	00:07:02
Random	0.0002	0.0001	0.0001	00:09:09

6.10.2 Experiment 2. Chronological split

The second experiment was conducted maintaining the cold-start issue in the data and splitting the datasets chronologically using the expanding time window approach described in Section 6.5. The set partitioning was performed on a monthly basis for D1 and on a bi-weekly basis for D2. The test set contained the last month of data for D1 and the last 2 weeks of data for D2. The training sets were of increasing length from

1 to 9 months for D1 and from 2 to 12 weeks for D2. The different partitioning scales were chosen because of considerably shorter timespan of D2 (even though it contained much more events). Figures 6 and 7 show the performance of algorithms for varied history lengths. Purple dots indicate peaks in performance.

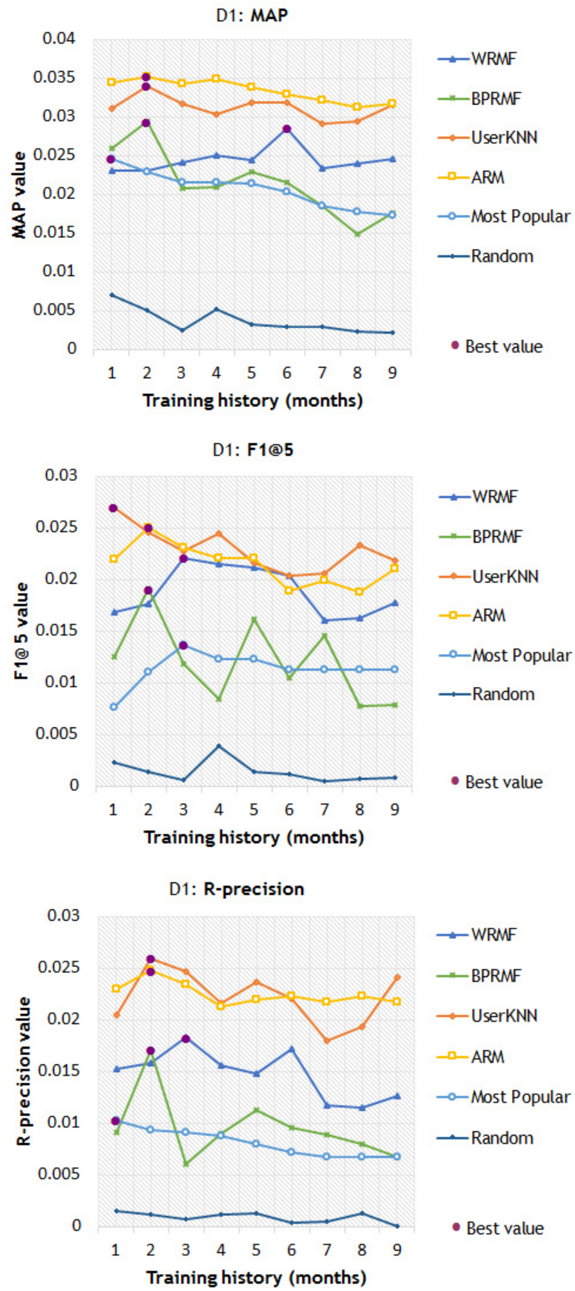


Figure 6: Chronological split of D1 with 9 training sets

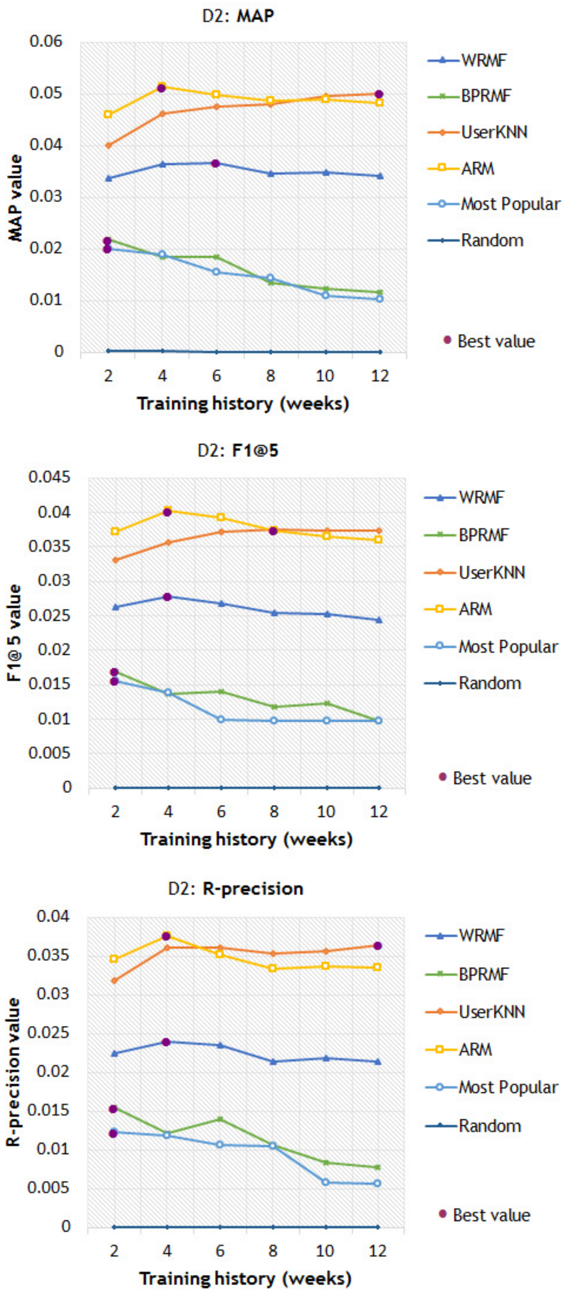


Figure 7: Chronological split of D2 with 6 training sets

From observing the performance peaks in Figure 6, it becomes clear that the optimal training history for D1 lies within the 3 recent months, which appears logical for a fashion store. A similar trend of favoring recent events is observed in Figure 7, with the exception of *UserkNN*, whose performance increases proportionally with the size of the history. It can also be seen that best-seller recommendations are generally most effective when recommending the most recently sold merchandize. These observations suggest the adoption of time-aware recommenders in e-commerce, which have shown improved quality of predictions in the movie ratings domain [154].

Table 10: Average scores for the chronological split on D1

Recommender	*MAP*	*F1@5*	*R-prec*	*Running Time*
WRMF	0.0244	0.0188	0.0147	00:05:27
BPRMF	0.0214	0.0121	0.0095	00:00:06
User*k*NN	0.0312	**0.0229**	0.0222	00:00:06
ARM	**0.0334**	0.0214	**0.0225**	00:00:01
Most Popular	0.0206	0.0113	0.0080	00:00:00
Random	0.0037	0.0014	0.0009	00:00:00

Table 11: Average scores for the chronological split on D2

Recommender	*MAP*	*F1@5*	*R-prec*	*Running Time*
WRMF	0.0351	0.0260	0.0224	02:27:10
BPRMF	0.0160	0.0130	0.0114	00:18:37
User*k*NN	0.0469	0.0363	**0.0352**	00:50:14
ARM	**0.0488**	**0.0377**	0.0346	00:20:35
Most Popular	0.0150	0.0114	0.0094	00:10:24
Random	0.0001	0.0000	0.0000	00:14:42

Tables 10 and 11 show the performance of algorithms after averaging their accuracy scores over all training sets. The results indicate that in general, time-based splits preserve the same ranking we observed in random splits, although with less significant difference in performance scores. The low measurement values is the consequence of cold-start conditions: predicting purchases for customers with short profiles is difficult (but realistic). However, this is not the only reason. We extended our experiment 2 by keeping test customers with at least 10 purchases. We then performed a chronological split followed by a 50/50 random split (as in experiment 1). The effect we observed after evaluating algorithms in these two protocols was quite remarkable: randomizing events in customer profiles boosted the performance of some algorithms by more than 300% (e.g. for WRMF). This observation tells that preserving the sequence of events during evaluation is crucial and that traditional random splits may produce too optimistic estimates of a recommender's accuracy.

6.10.3 Survey responses

31 responses were obtained in total from 20 commercial and 11 open source e-commerce platforms[21]. The responses to each of the 3 questions are summarized in Figure 8. Below we discuss some of the interesting findings.

Transactional data is used by more than 80% of respondents, which points to the clear dominance of CF over the content-based approach. At the same time, timestamps of events are neglected by most companies, which seems unreasonable in the light of our experimental findings. Other contextual sources of data have also shown low usage, even though more than a third of recommendation engines have been reported to be context-aware. This must be due to the comparatively higher percentage of ad-hoc recommenders based on domain knowledge.

[21] Among participants who chose not to stay anonymous were VP-Cart, Ecwid, WebJaguar, YesCart, NitroSell, S3Cart, Graphflow, NationKart, Whirlwind eCommerce, and PinnacleCart

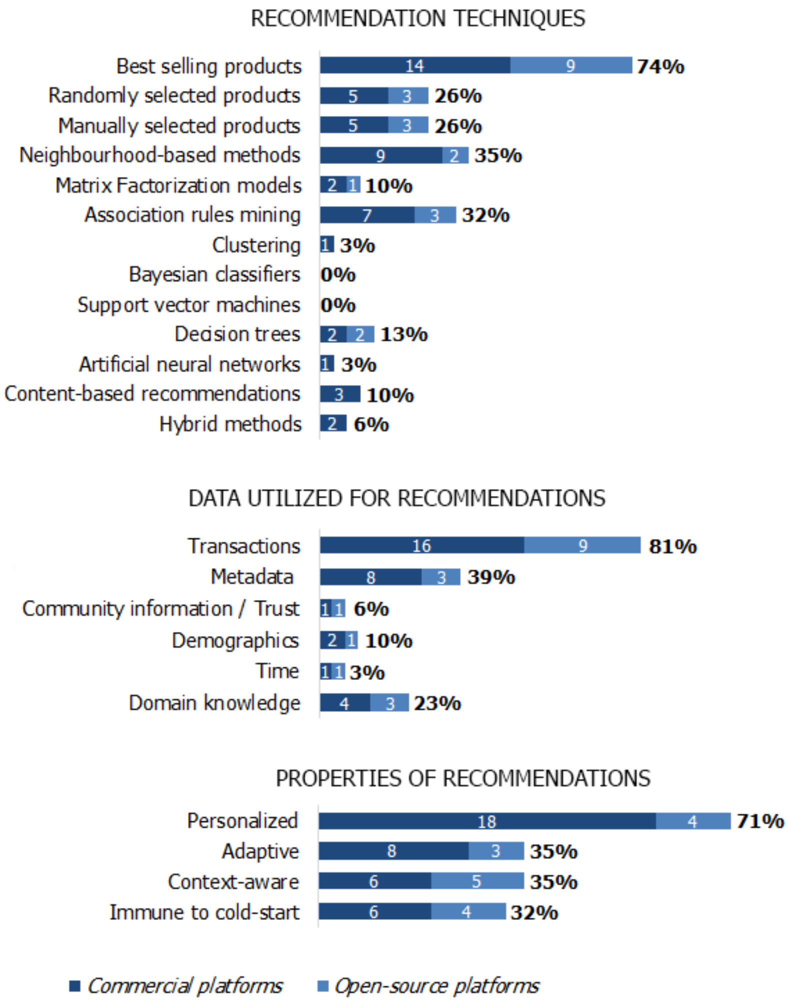


Figure 8: Survey responses

It seems that commercial platforms tend to put much more effort in personalization than their open-source competitors. We observe that neighborhood-based filtering and association rules mining are the most popular personalized recommenders, which totally conforms to our experimental results.

Recommending *best selling* products is the preferred approach for most platforms. Indeed, our experiments show that outperforming best-seller lists can be challenging even to personalized recommenders.

Furthermore, trivial methods of recommending *random* or *manually selected* products are more widespread than state-of-the-art methods, such as MF.

It appears that some platforms can be opposed to RS as such. One of the respondents argued that "if a shopper goes to all the trouble to choose the right product, it is almost insulting to link them to a page that says 'if you don't like this, then here are some recommendations.' Adding recommendations to a page that is the end result of a buying decision is a distraction and is likely to lower the rate of conversion".

6.11 Conclusions

Recommender systems is an important area of academic research and a vital revenue-generating tool in e-commerce. In this paper, we raised the question of the adoption of techniques in industry and the effectiveness of recent machine learning models for typical e-commerce settings, where explicit ratings are inexistent and implicit information is sparse.

We have shown that deployed e-commerce platforms take rather simplistic approach to product recommendations, with best-seller lists being at the heart of their engines. When it comes to personalized recommendations, the incline toward "good old methods" is not as surprising as it seems: our experiments confirm the superior performance of kNN collaborative filtering and association rules mining to more advanced MF-based methods when applied on binary purchase data. Similar findings have been reported in [230].

Whereas in academic context researchers tend to prioritize accuracy over speed (according to [20], only 11% of RS research papers report runtimes), in industrial settings where real-time interaction with the system is employed, the time factor is undoubtedly vital. This might be another reason in favor of simpler techniques. We show that time is also important in terms of the recency and the sequence of events. Depending on the type of merchandize, e-commerce datasets exhibit varied effects of seasonality. Thus, our experiments indicate that training on all available history is in most cases not optimal. We present a time-based splitting strategy that helps to establish the optimal timespan of a training set. This optimization can lead to improved recommendation accuracy and runtime efficiency. We also show that traditional random sampling, which often implies training on future events to predict past events, is prone to producing too optimistic performance estimates due to capturing erroneous shopping patterns.

Finally, we propose fast Golden Section Search for automatic tuning of algorithms with multiple hyperparameters.

6.12 Acknowledgment

We express our gratitude to Apptus Technologies for the provided datasets and computational resources.

7 PAPER II

Paraschakis D.

“Towards an Ethical Recommendation Framework”

Abstract The goal of our study is to provide a holistic view on various ethical challenges that complicate the design of recommender systems (RS). The articulated ethical recommendation framework maps RS design stages to the corresponding ethical concerns, and further down to known solutions and the proposed user-adjustable controls. The need for such a framework is dictated by the apparent lack of research in this particular direction and the severity of consequences stemming from the neglect of the code of ethics in recommendations. This framework aims to aid RS practitioners in staying ethically alert while taking morally charged design decisions. At the same time, it would give users the desired control over the sensitive moral aspects of recommendations via the proposed “ethical toolbox". The idea is embraced by the participants of our feasibility study.

Published in Proceedings of the 11th International Conference on Research Challenges in Information Science (RCIS), IEEE, 2017. *Best paper award.*

7.1 Introduction

In the era of Big Data, when the vast information overload makes it challenging for consumers to locate the products or services they need, *recommender systems* (RS) offer a helping hand in filtering the data. The notion of *recommendations* is built on real-life experiences and therefore perceived by humans as something inherently positive. User studies indeed show that the mere fact of labelling items as "recommendations" increases their chances of being consumed [51, 145]. Whenever this fact is exploited for reasons beyond serving user needs, an ethical problem arises. Formally, *ethics* can be defined as "the study of morality" [283], where *morality* in turn can be defined as "the system whose purpose is to prevent harm and evils" [89]. As we show later, in the context of recommender systems these "evils" may include privacy intrusion, identity theft, behavior manipulation, discrimination, offensive / hazardous content, misleading information, etc. Thus, we can define *recommendation ethics* as the study of the moral system of norms for serving recommendations of products and services to end users in the cyberspace. This system must account for moral implications stemming from both the *act* of recommending per se, and the enabling *technologies* involved. A holistic view on recommendation ethics is currently lacking in the field of RS, despite the massive research that it attracts nowadays. According to the recent study by Tang & Winoto [282], there exist only two publications ([245, 270]) that specifically address the problem of ethical recommendations. Still, they only focus on particular problems in particular applications. Concerns over the lack of holistic approach have also been brought up by Friedman et al. [82] who studied the privacy aspects of recommender systems - the subject that has drawn the most attention in the ethical discourse around the practices of big data. Another recent paper by Koene et al. [147] points to the striking research imbalance in the area of personalized RS. The authors note that the strong emphasis on the commercial success of recommender systems contrasted with the considerable neglect of moral values, has a potential risk of a future public backlash against this research area. Our study is also in line with the rapidly growing attention to Fairness, Accountability, and Transparency (FAT) in machine learning.

Recommendation ethics is a multifaceted problem, which relates to several interconnected topics that we broadly group into the ethics of *data manipulation* (privacy, anonymity, censorship issues), *algorithm design* (algorithmic biases, behavior manipulation, discrimination issues), and *experimentation* (fairness, awareness, informed consent issues). In Section 7.2, we touch upon all these topics in relation to recommender systems. In Section 7.3, we outline an ethical recommendation framework that serves two purposes: a) it provides a roadmap for an ethics-inspired design of RS; b) it proposes a toolbox for manual tuning of morally-sensitive components of RS. We evaluate our proposal in Section 7.4 by analyzing the results of the conducted survey. Section 7.5 concludes our work.

7.2 Theoretical background: ethical challenges

This section discusses various ethical challenges around recommender systems that have been identified in the related literature.

7.2.1 Data collection and filtering

Both types of personalized information filtering used in recommender systems - *collaborative* and *content-based* - require the construction of a *user profile*, expressed in either item attributes (metadata), or user interactions (ratings, purchases, etc). Unique user profiles make it possible to tailor recommended items to user's individual preferences.

Collecting behavioral data is often done in the absence of *informed consent*. Public surveys point to the great demand for “do not track" tools that would help users gain control over the data collection process [74]. Often, a notice about data logging is hidden inside the Terms of Service (ToS) [147], which users are expected to read and agree with. However, the well-familiar line *“I have read and agree to the terms”* has been coined as “the biggest lie on the Internet" [212] due to the fact that people either do not read or do not understand these lengthy and legalese-heavy notices. Even if they did, they would still have to spend

40 minutes per day on average for doing so [185]. A number of initiatives[22] have been launched to improve the current situation by standardizing ToS, introducing trust marks or centralized monitoring. In Europe, the EU Cookie Law attempts to strike a compromise between brevity and informativeness by enforcing minimalist pop-up banners for acquiring user consent to store cookies. Sadly, both approaches fall victim of a "click to close" attitude from internet users [147]. It is worth noting, however, that the legislation pertaining to data collection, use, and disclosure has become more comprehensive in recent years with the introduction of the U.S. Consumer Privacy Bill of Rights and the European Privacy Directive [82].

To improve the accuracy of recommendations, user profiles can further be enriched by gathering additional data from external sources. This can be achieved by means of tracking cookies [246], linked open data [103], social networks [55] and even information brokers who collect and resell user activity data [212]. Whereas exceptionally detailed user profiles have positive effect on personalization, their leakage and misuse can put user's privacy at risk. Friedman et al. [82] distinguish three major sources of privacy breaches in recommender systems: a) the *recommendation engine* itself (and the people behind it), when its operation does not live up to users' individual expectations of privacy; b) *RS users*, who may attempt an inference attack on each other's profiles by observing RS outputs; and c) *external adversaries* (e.g. hackers), who may attempt to re-identify subjects in anonymized data (see more in Section 7.2.2).

When fallen into the wrong hands, behavioral profiles built for serving personalized recommendations may as well be utilized for malicious purposes, such as *phishing* or *social engineering* [147]. Moreover, disclosed user profiles may reveal sensitive private information either directly (e.g. health records) or indirectly (e.g. by inference from movie ratings). *Profile injection* is one of the possible attacks, which uses fake profiles to promote or demote recommendations of certain items [35]. For example, it can be used to suppress recommendations of com-

[22] `http://www.commonterms.net`, `http://tosback.org`, `http://www.truste.com`, `https://wiki.mozilla.org/Privacy_Icons`

petitor's products by giving them low ratings from a fake profile. To tackle this problem, some websites (e.g. Booking.com) enforce user identity verification as a prerequisite for leaving feedback. The emergence of *recommendations-as-a-service* (RaaS) in the past few years has also raised privacy concerns due to the exposure of user profiles to a third party [82].

A number of *privacy preserving collaborative filtering* (PPCF) techniques have been proposed in RS literature (e.g. [165, 229, 320]) to protect user profiles from misuse. These solutions can be broadly classified into *architectural* (protocols, certificates), *algorithmic* (cryptography, differential privacy, pseudonymity, obfuscation), and *normative* (policies, regulations) [82]. The major challenge in such privacy-enhanced RS is to preserve the recommendation accuracy at an acceptable level. A comprehensive overview of privacy aspects of RS can be found in [82]. It is therefore crucial that RS developers become aware of potential privacy implications and take the necessary precautions during the initial data gathering/user profiling stage.

Apart from employing privacy-enhancing technology (PET), recommender systems can benefit from graphical user interfaces (GUIs) that enable users to manage their profile data by defining interests, cleaning up browsing history, and so on (e.g. see Amazon's "manage history" tool). Bakalov et al. [15] show that this has a positive effect on RS usefulness and usability, as well as on the overall user satisfaction. The reason is that users have different perceptions of privacy and different attitudes towards the associated protective measures [22]. To this end, *user-tailored privacy decision support* for recommender systems has received considerable attention in the literature [144, 145, 306]. Despite the provided ability to manually withhold items from one's profile history, there is a danger that these items may still be inferred from other information exposed to a recommender system [82]. Therefore we advocate the use of PET-based and GUI-based solutions in combination as a complex protective measure.

The final stage of information filtering in a RS delivers relevant items to a user. With very few exceptions ([245, 270, 282]), existing systems do not employ any ethical filtering of their output. This raises an issue of content censorship with all its consequent implications. As well noted in [282], what is algorithmically appropriate is not necessarily ethically appropriate. The serendipitous nature of a collaborative RS adds an extra challenge due to its tendency to produce unexpected results. Apart from the obvious need for parental control, one might think of morally troubling examples of recommending meat-based products to a vegetarian, gay-intolerant movies to a homosexual, alcohol drinks to a religious Muslim, or tobacco products to a person who struggles to quit smoking. Should the recommender system be held accountable for serving offensive or even hazardous content? How can it mitigate the risk of moral harm? The ethical appropriateness of candidate items can be established by mapping potentially harmful elements in media content (drug use, nudity, etc.) to a user's persona (gender, age, religion etc.), as explained in [282]. The authors suggest that moral values are incorporated into system requirements, with users having control over the filtering process. We add that moral values can be treated as a special type of *context*, which can be used in pre-, post-, or in-filtering stages of recommendations. An attempt to build a fully-automated ethical RS is made in [270], where the human intervention is not required. Yet, the reliability of such a system largely depends on user-specified demographic data, which can often be omitted or falsified by users due to privacy concerns. Besides, applying an automatic censorship filter in the complete absence of user control (and consent) would also be morally problematic. Facebook's profanity filter for blocking offensive words serves as a good example of a user-adjustable censoring tool.

7.2.2 Data publishing and anonymization

The active research on recommender systems would hardly be possible without the availability of publicly released datasets from online services such as Netflix, MovieLens, or LastFM. Annual RecSys challenges and contests such as the Netflix Prize propel research and help businesses improve their recommendation engines by means of crowdsourcing RS algorithms from a large academic community. Because a *public* dataset contains *private* data [188], releasing a myriad of user records bears serious moral responsibility. The risks of disclosing a user profile were mentioned in the previous section. Here, we continue this discussion from the perspective of *privacy preserving data publishing* (PPDP).

For several decades now, *anonymization* has been used to disguise personally identifiable information (PII) in public datasets by employing such techniques as generalization, suppression, perturbation, or anatomization [311]. Barocas et al. [18] define *anonymity* as the state of complete unreachability, whose absence would pose a threat to one's ethical and political freedoms. The naive assumption that such anonymity can be achieved by disguising explicit PII such as names or phone numbers has long ceased to hold due to the existence of *quasi-identifiers*. When combined, these seemingly harmless fields are capable of uniquely identifying a user (e.g. 87.5% of U.S. residents can be identified via a $\langle$*ZIP code, birth date, gender*$\rangle$ combination [276]). However, user demographics is a valuable input for many recommenders, and a too aggressive anonymization may impact the usability of a dataset (especially for a content-based RS). This challenging trade-off between privacy and utility lies in the core of PPDP. Taking an online pharmacy dataset as an example, its public release might go either of the two extreme ways, both leading to a highly undesirable outcome:

- heavy anonymization $\longrightarrow$ inaccurate recommendations $\longrightarrow$ purchase of incompatible medications

- poor anonymization $\longrightarrow$ re-identification of subjects $\longrightarrow$ disclosure of sensitive health records

Data owners are therefore encouraged to perform a *privacy calculus* [145] before deciding on whether and how a public release should be administered. This is done by "multiplying" the risk of disclosure by the sensitivity of the data [213], weighted against the anticipated benefits. Such privacy calculus can potentially be automated, as explained in [145]. If data sensitivity is domain-dependent, should there be regulations dictating how to treat data from a specific domain before the release? Should there be different levels of domain-specific data access for different types of public (researchers, students, data brokers, advertisers, etc.)? Corporate ethics sometimes result in *semi-public* releases, when the dataset's accessibility by external analysts is bounded by the physical organizational boundaries. Notably, anonymization may become a necessity even on platforms with a *closed* recommender system [103], which operates on a privately maintained dataset. This is done to protect user identities when the data is shared between different departments of a large organization [213].

Notably, various anonymization techniques aimed at protecting privacy in public data are ethically problematic in themselves. This is because obfuscated user profiles can be interpreted as "invalid data", which may potentially lead to incorrect inferences about individuals (even if the recommendation accuracy remains high). This may have negative psychological and even legal consequences [82].

The possibility of aggregating data from external sources unlocks the door to easy re-identification even in the presence of substantial anonymization. This can be done through a *linking attack* [176], which matches records in different databases via quasi-identifiers. This attack was effectively used to de-anonymize the Netflix dataset (containing 500,000 users) only two years after its public release [203]. As a result, the planned Netflix prize sequel was discontinued due to the filed lawsuit [147]. It was shown in [203] that with some minimal and imprecise outside information about the user (a couple of public ratings from IMDb and their approximate dates), it was possible to identify him/her in the Netflix dataset with high probability. Apparently, disclosed movie ratings can be used to infer sensitive facts about a person, such as their political/religious views, or sexual status. Moreover, revealing movie rating history may have implications on user's *future* privacy [203]. Naturally, this applies to other types of preference data too (music tastes, Facebook "likes", etc.). The risk of privacy breaches is further exacerbated because of the *accretion problem* [213], which refers to the snowballing effect of chained linking attacks on the increasing number of databases. Linking large chunks of external data has become easier than ever thanks to the rise of big data, semantic web, and social networks. The case of de-anonymizing the completely PII-less Twitter graph by intersecting it with the social graph of Flickr serves as an instructive example [204]. Ethical concerns about de-anonymization of data might explain the limited number of public datasets available for RS research.

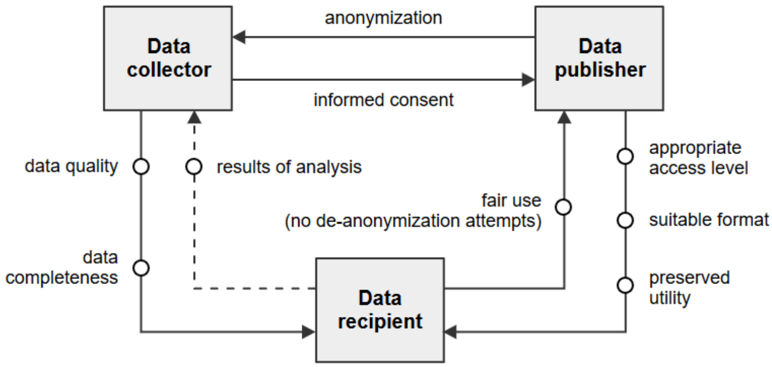


Figure 9: Moral bonds between stakeholders in the data publishing process

Figure 9 depicts moral bonds between the three main stakeholders in a data publishing cycle, namely *data collector*, *data publisher*, and *data recipient*. These bonds reflect the moral responsibility of stakeholders towards each other, as well as the data subjects involved. The responsibility of the data collector is to ensure that the data sample is representative of the population, clean, and contains the necessary attributes. The consent is then given to the data publisher to proceed with the release. In return, the data publisher guarantees anonymity and privacy protection of data subjects. In front of the data recipient, the publisher is responsible for making the dataset accessible (in compliance with the active policies), properly formatted, and usable for analysis. The data recipient is morally entitled to the fair treatment of data and - in some cases and upon agreement - communicating the results of the analysis back to the data collector/owner. We note that this is just an idealized framework, whose practical implementation is currently confronted with challenges of technological or legislative nature.

While privacy models are being developed by computer scientists to aid PPDP (differential privacy [69], *k*-anonymity [277], *l*-diversity

[176], etc.), lawmakers fight their own battle to regulate data publishing and catch up with the ethical realities of big data analytics. At the very least, legislators must acknowledge that the traditional anonymization paradigm based on the removal of PII is nothing more than a half-measure [213]. We refer the reader to [213] for an in-depth discussion of the regulatory challenges concerning PPDP.

7.2.3 Algorithmic opacity, biases, and behavior manipulation

A typical recommender system operates as a black box, with its *output* being the only part that is visible to a user. Whereas certain *input* signals can also be observed (e.g. product attributes, personal purchase record, public ratings), aggregating and processing them into recommendations is done completely behind the scenes (Figure 10).

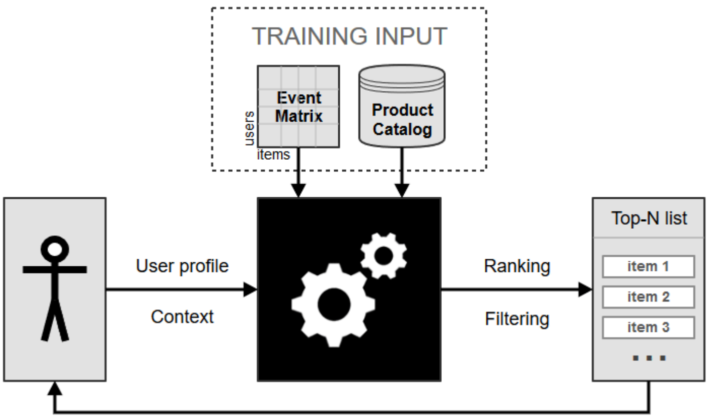


Figure 10: High-level view of a personalized recommender system

What are the ethical implications of not knowing the *algorithm* that rules recommendations? Apparently, this opacity makes it hard to reason about the political, economic, and cultural agendas behind these suggestions [223]. It can also hide the evidence of *algorithmic discrimination* [253] and *behavior manipulation* [147]. As Pasquale [223] notes, "faulty data, invalid assumptions, and defective models can't be corrected when they are hidden". This is why algorthmic opacity is viewed as a serious ethical issue in data science. Should users be allowed to look inside the black box? The survey by Herlocker et al. [105] reveals that 86% of users would like to see an *explanation interface* as an integral part of the recommendation service. The simplest example is Amazon's "customers who bought this item also bought..." recommendation interface. Automatic explanations can provide transparency into *how* and *why* a recommendation was generated, and hence improve users' trust in the system [105, 285]. Especially when a recommender system gives bad advice, users deserve an explanation and a possibility to correct the system's reasoning (this property is known as "scrutability") [285]. One way of aiding transparency is to report the system's *confidence* in generated recommendations.

Whereas explanations offer a way to unbox a recommendation engine, they also have a number of ethical implications. This practice often comes in conflict with corporate ethics that may forbid the disclosure of an algorithm for fear of losing competitive advantage or making it vulnerable to intentional manipulation [61]. The challenge in this case is to find other ways to justify recommendations without having to open the black box. Some examples of such justifications are given in [105]. Other alternatives proposed in the literature include routine *algorithm audits* by third parties [253], and *reverse engineering* methods [61]. Perhaps less obvious, there also exists a privacy threat that exploits the fact that explanations reveal connections between items. Combined with some side information, these connections would allow an adversary to identify a specific person and reveal their personal details [236]. Finally, the provision of transparency is impeded because of the *interpretability* problem [37]. Even if an algorithm is not kept in secrecy, explaining it to a user may hardly be possible because of its inherent mathematical complexity. For example, some recommendation methods are based on *latent semantic analysis*, where the distinguishing factors do not have a clear interpretation [285]. The problem is further exacerbated by the "course of dimensionality", which makes it impossible for a reasoning human to grasp the extreme multitude of *features* involved in computation [37]. Besides, unveiling the algorithm in this case would not be sufficient, since predicting its behavior without knowing the specific feature *values* is virtually impossible [253]. Finally, some of the most successful recommender systems are implemented as hybrid and ensemble models, combining several algorithms and data sources. Comprehending the reasoning behind such complex systems may be very hard even for trained computer scientists.

In their interaction with RS, how can users be certain that their interests are respected and prioritized? What if the recommendation output is severely biased and their behavior is being intentionally manipulated to meet certain business objectives? Documented examples of algorithmic bias include recommending lower paying jobs to female candidates, or higher priced flights to MacBook owners [243]. As Ross [247] notes, there is a big difference between using a dataset for *describing* a population and using it for *manipulating the behavior* of that population. For instance, this can happen when a recommender system is exploited to push expiring, overstock, de-listed, and other first-priority items that do not necessarily answer the needs of a user. It may be tempting for a business to disguise such items under misleading category names such as “best sellers" in order to boost the utilization of any unwanted inventory. In a similar way, privileged recommendation categories (e.g. “top hotels") can be easily populated with paid listings that may not correlate well with the category label.

Another type of bias that can be encountered in recommender systems is *price discrimination*. This enables a RS to adjust prices on the fly based on the user's *willingness-to-pay* (WTP) value, defined as the maximum amount of money s/he is expected to pay for a given product. It has already been shown that travel sites Cheaptickets and Orbitz intentionally raise hotel prices for visitors who do not have an account with them [99]. Another example is Amazon's short-lived experiment on selling DVDs at different prices to different users, which provoked a furious reaction showing the profound moral impact of price discrimination [292]. Nevertheless, this did not stop researchers from proposing recommendation models with personalized pricing (e.g. [129, 183, 325]). Unfortunately, the ethical feasibility of such systems remains poorly addressed and hence questionable, especially considering that price discrimination is illegal in some jurisdictions. Even when it is not, this serves as a good example of when legal does not equate to ethical. An even worse case can be made when a discrimination is based on morally sensitive attributes (gender, race, religion, etc.) that are either directly provided by a user, or inferred from other information available to a recommender system (as discussed earlier). For example, when a person is to be recommended for a loan, gender-discriminatory decisions may introduce dangerous "false negatives" that may potentially lead to lawsuits [295]. Such systems can (and should) benefit from specialized techniques for *discrimination-aware data mining* [76].

Personalized news recommenders, to a large extent, can be accountable for the creation of the so-called "filter bubbles". This dangerous side effect of personalization turns news consumption into a one-sided propaganda by serving stories that only reaffirm one's biases. This issue came to the forefront with a vengeance after the recent unexpected election of the U.S. president Donald Trump, which was largely attributed to the dissemination of fake news over the internet [8]. According to Chaslot & Gorbatai [46], 80% of recommended videos on YouTube favored Trump over Clinton (regardless of the posed queries), most of which featured false stories.

How can users gain control over the aforementioned biases introduced by the RS providers? Is it users' moral right to receive recommendations that best match their interests? The freedom to choose what criteria the recommender system is optimized for may increase user satisfaction and trust through better personalization. We therefore believe that the RS's marketing bias should become adjustable by users, giving them the ability to reset the system's current optimization goals to the default, or "best match", mode. To provide evidence that a such a system is indeed free from biases, it can expose explanation interfaces, as discussed earlier. However, to fully address the problem of behavior manipulation, a regulatory oversight by trusted third parties is advised [147].

7.2.4 Online experiments

RS practitioners willing to assess the real effect of a new recommendation algorithm (or modifications of an existing one) often embark on *A/B testing*. This is an online between-subjects experiment, where control and treatment groups are assigned the *original* and the *modified* variants of an algorithm, respectively. Digital businesses ranging from small start-ups to internet giants such as Yahoo, Twitter and Ebay, continuously run online experiments in attempt to improve their services. In the vast majority of cases, however, users are silently dragged into these experiments without their knowledge, let alone consent. Taken to an extreme, the exposure to a previously untested algorithm is somewhat akin to medical trials, where the consequences of applying a new method may lead to unpredictable side effects and potential harm.

It is worth mentioning two (in)famous examples of online experiments, which have clearly stepped over the ethical line. In 2012, Facebook studied the phenomenon of emotional contagion by intentionally manipulating the moods of more than 600,000 users through their news feeds. Two years later, dating service OKCupid studied the emotional effect of recommending bad partner matches to its users while presenting them as "exceptionally good". Both experiments failed to acquire explicit permission from participants, while toying with their emotions and private lives. This gave rise to a public uproar and debates about the ethics of A/B testing. Referring to the aforementioned cases, Grimmelmann [93] reminds that running experiments on people is regulated by laws and as a minimum, requires the provision of informed consent and approval of an Institutional Review Board (IRB). Researchers embarking on A/B testing need to establish whether the data protection laws fall under the jurisdiction of countries where researchers or participants reside, where the server or the IRB are located, where the data is analysed, or some combination of these [247].

There are several ways of dealing with informed consent in online experiments. In the "opt-in" scenario, user consent is acquired explicitly via a prior notice. Given the ubiquity of online experiments (we note the ubiquity of RS as well), implementing prior consent in practice does not appear particularly realistic. Furthermore, prior consent introduces an unavoidable bias that may potentially influence the results and make the sampling less than random [231, 247]. Another possibility is the "implied consent" acquired through ToS, with an added option to opt-out from experiments at any time. Implied consent may be sufficient for experiments with low risk [78]. Since many companies perform A/B testing with no consent at all, the "notify after" approach can be taken to retroactively inform users about the performed study, providing them with the results, contact information of the involved researchers, and offering the option to delete their data [231]. Felten [78] warns that this alternative is only applicable to situations with minimal risk. Thus, the perceived moral impact of the experiment should be a deciding factor for obtaining informed consent. It is important to remember that when the risk of moral hazard is high, the test is unethical even with consent from participants [78].

When a recommender system is A/B tested for potential gains from applying an experimental algorithm, the two variants of algorithms (i.e. original and experimental) may differ significantly in their output. For example, hotel booking operators Expedia and Hotels.com are known for conducting A/B testing experiments that steer one of the user groups towards more expensive hotels [99]. This raises the question of fairness: is it users' moral right to demand the interaction with the trusted ("default") system instead of the one whose behavior is far less obvious? Again, we suggest that the ability to reset the recommender system to its default algorithm is provided as an option to RS users. Considering that A/B testing is undoubtedly an indispensable evaluation tool in a RS developer's arsenal, it is important to ensure that it is practiced in an ethical manner.

7.3 Summary as a framework

In this section we summarize the findings of the study in the form of a user-centric ethical recommendation framework. The framework maps various RS development stages to potential ethical concerns and the recommended countermeasures. As a practical contribution, we propose an "ethical toolbox" comprised of user-adjustable controls corresponding to each development stage. These controls enable users to adjust the recommender system to their individual moral standards. Table 12 outlines the articulated framework, which also details the meaning of each component of a toolbox.

Table 12: General user-centric ethical recommendation framework

Development stage	**User profiling** (input stage)	**Data publishing** (optional sharing stage)	**Algorithm design** (modeling stage)	**User interface design** (presentation stage)	**Online experiments** (optional testing stage)
Ethical concerns	Privacy breaches, lack of awareness / consent, fake profile injection	Privacy / security / anonymity breaches	Biases, discrimination, behavior manipulation	Algorithmic opacity, content censorship	Unequal treatment, side effects, lack of trust / awareness / consent
Known countermeasures	Informed consent, privacy preserving collaborative filtering, user identity verification	Privacy preserving data publishing	Algorithm audits, reverse engineering, discrimination-aware data mining	Explanations, ethical rule set generation, content analysis	Informed consent, possibility to opt out and delete data
User-adjustable controls (the toolbox)	"Do not track activity" tool *This setting restricts the creation and the maintenance of a user profile. Types of data can be manually defined and browsed items can be manually deleted (e.g. "manage history" tool on Amazon)*	"Do not share data" tool *This option allows local user profiling but restricts data sharing with third parties (even in the presence of anonymization). Types of data or categories of allowed recipients can be manually defined*	"Marketing bias" filter *This filter is used to remove any business-driven bias introduced by RS providers, and sets the recommendation engine to the "best match" mode (or other user-selected modes, such as "cheapest first")*	"Content censorship" filter *This tool can be used to set user-defined exclusion criteria for filtering out inappropriate items or categories. It also contains the option to turn the filter on and off (also with the possibility of scheduling)*	"Opt out of experiments" tool *This option can be used to reset the recommendation engine to its default algorithm, exclude the user from any future experiments, enable the opt-in option, and delete data from previous experiments*

The provided transparency and control over recommendations are known to increase users' trust in the system [105, 145]. At the same time, empirical studies show that users tend to avoid the hassle of adjusting settings once they get the ability to do so [145]. However, we argue that the existence of the proposed controls is what makes the system *ethics-aware by design* and thus helps to retain prudent RS users who would stop using the system otherwise. One way of reducing the burden of adjusting settings is to implement *adaptive defaults*. For example, the "do not track activity" control can be pre-set according to the user's expressed willingness to accept cookies, or according to the estimated sensitivity of information involved in user profiling. Likewise, the content censorship filter may automatically enable itself during periods when children are expected to be at home. Certain controls, such as "do not track activity" and "do not share data" can be combined to simplify user interface. In general, the usability of the provided controls may depend on many factors, such as their layout, frequency of using the system, sensitivity of data, and so on. As a vital first step, however, it is necessary to establish the general stance of users towards the ethics of recommender systems and whether the proposed toolbox would stand as a viable solution. This is done in the next section.

7.4 Feasibility study

We conduct an online survey[23] to find out people's opinions and their preferred course of action regarding five ethical aspects of recommender systems that the proposed toolbox aims to address, namely: *user profiling, data sharing, marketing bias, content censorship*, and *online experiments* (corresponding to the five stages of the framework in Table 12). The survey was disseminated to Facebook groups of numerous European universities, yielding 224 responses from students and academic staff. 10 responses from people who claimed to be totally unfamiliar with recommender systems were filtered out, leaving us with 214 valid opinions. Instead of directly asking whether participants would be willing to have a manual configuration tool for a particular ethical

[23] available at `http://recommendations.typeform.com/to/kgKNQ0`

Table 13: Issue 1/5: User profiling

Q1.1. **Which of the following factors, if any, would make you give up on personalization in favor of preserving privacy?** (multiple answers allowed)	**# (% cases)**
A) Ability to infer sensitive information from my browsing behavior, such as my sexual status, health condition, or religious beliefs.	139 (65.0%)
B) Revealing my browsing behavior to other users of the same account (for example, other family members using the same website)	116 (54.2%)
C) Using my profile for targeted advertising campaigns / promotional emails	111 (51.9%)
D) None of the above. I want to receive personalized recommendations and I am not concerned about privacy	11 (5.1%)
E) Other	7 (3.3%)
F) I don't care either about recommendations or privacy when I browse the web	6 (2.8%)
Q1.2. **How should user profiling be handled?**	**# (% cases)**
A) Recommendation engines should provide user-adjustable controls for setting user profiling on/off manually	169 (79.0%)
B) Agreeing to the Terms of Service and allowing/disallowing cookies on websites makes me feel safe about my user profile	38 (17.8%)
C) I don't care about user profiling, and I don't want to be informed or asked about it	7 (3.3%)
D) Other	0 (0.0%)

issue, we offered them a choice between several alternative options. The results of the survey are summarized in Tables 13-17, where answers are listed in descending order of the number of votes received. To mitigate ambiguities, each question was preceded with a short explanation of a particular ethical aspect (omitted from Tables 13-17 for brevity). We note that some complementary questions allow multiple answers (hence percentages do not always sum up to 100).

Table 14: Issue 2/5: Data sharing

Q2.1. **Which of the following probable implications, if any, would make you refrain from sharing your anonymous profile, even if it would contribute to better recommendations in the future?** (multiple answers allowed)	**# (% cases)**
A) Hacker attacks on anonymized public data which may result in re-establishing my identity	123 (57.5%)
B) The possibility of using my data for purposes other than research	117 (54.7%)
C) The possibility of finding out additional information about me by matching records in different databases	116 (54.2%)
D) My willingness to share data depends on how much I trust the particular website that gives recommendations	96 (44.9%)
E) None of the above. I am not afraid of publishing my anonymized profile online as long as it benefits the service I get	10 (4.7%)
F) Other	1 (0.5%)
Q2.2. **What is your stance towards sharing your data with third parties for research purposes?**	**# (% cases)**
A) I want to be able to control data sharing myself in the settings of a recommendation engine	104 (48.6%)
B) I am willing to share my data with third parties, as long as it stays anonymous. This should be inscribed in the ToS	65 (30.4%)
C) I don't want my data to be shared with third parties under any circumstances, even if it has been anonymized	45 (21.0%)
D) I don't have an opinion	0 (0.0%)
E) Other	0 (0.0%)

Table 15: Issue 3/5: Online experiments

Q3.1. **Which of the following reasons, if any, would make you refrain from participating in an A/B testing experiment?** (multiple answers allowed)	**# (% cases)**
A) Lack of consent: the fact that I am not notified or asked for permission to participate in the experiment	103 (48.1%)
B) None of the above. I don't mind testing the new algorithm, which might work even better than the old one	94 (43.9%)
C) System instability: I cannot be sure that I will get the same quality of recommendation the next time I visit the website	30 (14.0%)
D) Lack of trust: I wouldn't trust an advice coming from an experimental, previously untested system	25 (11.7%)
E) Unfairness: it is not fair to treat the users of the same system unequally	16 (7.5%)
F) Other	1 (0.5%)
Q3.2. **What would be the preferred method of getting your consent for participating in an A/B testing experiment?**	**# (% cases)**
A) Contact me before the experiment and ask for my permission to participate	84 (39.3%)
B) Don't contact me at all, but provide an option in the recommendation engine for controlling whether I want to interact with the existing system or participate in testing of a new recommendation method	73 (34.1%)
C) Inform me after the experiment, but allow me to delete any data I generated during the experiment, if I want to	33 (15.4%)
D) Don't contact me at all, and let the system do what it finds appropriate	23 (10.7%)
E) Other	1 (0.5%)

Table 16: Issue 4/5: Marketing bias

Q4.1. **What do you think about the marketing bias of a recommendation system?**	**# (% cases)**
A) I don't expect salespeople of physical stores to be unbiased in their recommendations. Online recommendations are no different. If it's important for the business, I don't mind	90 (42.1%)
B) This is clearly unethical because I expect recommendations to be unbiased and tailored to my preferences at all times	81 (37.9%)
C) I don't mind the marketing bias because I may find such "biased" recommendations even more interesting	17 (7.9%)
D) I don't have an opinion	16 (7.5%)
E) Other	10 (4.7%)
Q4.2. **How should the marketing bias be handled?**	**# (% cases)**
A) Removing or allowing a marketing bias should become a user-adjustable option of a recommendation engine	80 (37.4%)
B) Companies have the right to adjust their recommendations as they please. Users should not intervene	60 (28.0%)
C) Recommendation engines should never be allowed to recommend anything that doesn't serve user needs	49 (22.9%)
D) I don't have an opinion	19 (8.9%)
E) Other	6 (2.8%)

Table 17: Issue 5/5: Content censorship

Q5.1. **How should content censorship be handled in recommendation systems?**	**# (% cases)**
A) Recommendation engines should have a user-adjustable censorship filter, where I can set my exclusion criteria and turn the filter on/off at my own will	104 (48.6%)
B) There should be no censorship. I want to be able to examine all relevant options and judge myself what is appropriate	58 (27.1%)
C) Recommendation engines must filter out inappropriate items automatically, based on my profile data or something else	31 (14.5%)
D) I don't care if there is any content censorship or not in recommendations	19 (8.9%)
E) Other	2 (0.9%)

The examination of survey results immediately reveals participants' strong preference for taking morally sensitive issues under their control. In 4 out 5 cases, the majority voted for having a user-adjustable setting within a recommendation engine for controlling a particular ethical issue. In the following paragraphs we analyze the responses (including the free-form ones where these were deemed relevant and insightful).

Online experiments (Table 15) was the only case where a user-adjustable option came second (however with only 5.2% difference from the top choice). Considering that the lack of consent was identified as the biggest ethical barrier to participating in A/B testing (Q3.1), the indicated preference towards the opt-in scenario (Q3.2) appears well justified, as it is linked with maximal risks. As one of the participants noted, requests from users to delete their data implicitly signals that they should best be excluded from future experiments.

When it comes to user profiling (Table 13), we observe a strong trend of preferring privacy over personalization. Users' major concern is the disclosure of their browsing behavior and the risks associated with it (Q1.1). In fact, 5 respondents stated explicitly that they would not trade their privacy for personalization for whichever reason, whereas one participant stated that s/he would only accept personalization on a group level, while staying anonymous within the group. Our results once again confirm the failure of ToS to satisfy users' privacy needs (Q1.2). Instead, the vast majority of users (79%) want to have the ability to control user profiling manually.

Exactly the same attitude was expressed by users towards sharing their data with third parties (Q2.2) due to various privacy concerns (Q2.1). Notably, almost half of users would adjust their data sharing strategy depending on how much they trust the particular website.

The marketing bias of RS (Table 16) brought up the most controversy as it can be seen in Q4.1, where two opposite views divided the overwhelming majority of users into two camps. Interestingly, the largest camp is ready to accept biased recommendations, with 7.9% of users feeling especially optimistic about them. The challenge of this particular issue can also be judged from the considerably large proportions of hesitant respondents who refrained from answering Q4.1 and Q4.2. Still, both camps prefer to have a manual control over the marketing bias in the settings of a recommender system. This issue also yielded a larger number of free-form responses, most of which starkly manifested users' negative stance towards recommendations because of the awareness of the marketing bias. One user also expressed their concern about getting falsely advertised recommendations (e.g. abusing explanations of type "other users also bought", etc.), which were mentioned in Section 7.2.3.

As far as the censorship is concerned (Table 17), the preference for being able to filter the recommendation output manually clearly dominated other options in Q5.1. On the other hand, the preference for having no censorship considerably outweighed the preference for automatic content filtering. Thus, having the former option as a "smart default" is well justified. In any case, censorship appears important to the overwhelming majority of RS users (91.1%).

7.5 Conclusion

We conclude that multiple moral dilemmas of varied severity emerge on every stage of RS development, while their solutions are not always evident or effective. In particular, there are many trade-offs to be resolved, such as user privacy vs. personalization, data anonymization vs. data utility, informed consent vs. experimentation bias, and algorithmic transparency vs. trade secrets. Balancing between consumer and corporate ethics in recommender systems is a challenging task. A careful assessment of moral risks is crucial for deciding on the strategies of data anonymization or informed consent acquisition for performing A/B testing or user profiling. Yet, the question of precisely who and how should perform such risk analysis is still a subject of debates. We have found evidence that many big players on the RS market (Facebook, Amazon, Netflix, etc.) have faced loud ethics-related backlashes. Thus, it is important to ensure that the RS architecture is not only legally and algorithmically justified, but also *ethically* sound.

The articulated ethical recommendation framework suggests new paradigm of *ethics-awareness by design* with an addition of a contextual layer that utilizes existing technologies where possible and complements them with user-adjustable controls. This idea was embraced by the vast majority of our survey participants, and future work should further test its viability in a fully implemented RS prototype.

8 PAPER III

Brodén B., Hammar M., Nilsson B.J., and Paraschakis D.

"A Bandit-based Ensemble Framework for Exploration/Exploitation of Diverse Recommendation Components: an Experimental Study within e-Commerce"

Abstract This work presents an extension of Thompson Sampling bandit policy for orchestrating the collection of base recommendation algorithms for e-commerce. We focus on the problem of item-to-item recommendations, for which multiple behavioral and attribute-based predictors are provided to an ensemble learner. In addition, we detail the construction of a personalized predictor based on k-Nearest Neighbors (kNN), with temporal decay capabilities and event weighting. We show how to adapt Thompson Sampling to realistic situations when neither action availability nor reward stationarity is guaranteed. Furthermore, we investigate the effects of priming the sampler with pre-set parameters of reward probability distributions by utilizing the product catalog and/or event history, when such information is available. We report our experimental results based on the analysis of three real-world e-commerce datasets.

Published in ACM Transactions on Interactive Intelligent Systems (TIIS), Volume 10(1), Special Issue "Highlights of IUI 2018", ACM, 2019

8.1 Introduction

Dynamically responding to user queries with top-N item ranking is a very common problem on the web, which pertains to search, advertising, and recommendations. A specific use case of interest is generating *item-to-item* (i2i) recommendations for every visited product page with the aim to maximize product sales. Such *non-personalized* recommendations are useful when user profiles are limited or nonexistent. This is very typical of online shopping sessions, where most visitors are unidentified [289, 308], yet keen to see meaningful recommendations already on their first page visit (for example on a landing page). Such dynamic environments with streaming event data and severe cold-start make conventional recommender systems impractical and call for *reinforcement learning* approaches, whose exploration-exploitation paradigm appears very appropriate [166].

Item rankings can be generated by various predictors that utilize item consumption patterns and/or content features. Learning which predictor (or a combination thereof) is optimal in a given context in an online manner naturally translates to a *multi-arm bandit* (MAB) problem. In its classical formulation, a policy sequentially chooses from a finite set of actions with stationary (but unknown) reward distributions in attempt to maximize the total reward of the policy (or equivalently, to minimize the total *regret* - the amount lost due to not choosing the optimal arm in each step). This problem has been extensively studied in the literature, especially since the seminal paper by Lai & Robbins [159]. They introduced upper confidence bound (UCB) policies attaining a logarithmic regret (see Section 8.8.5 for more details on UCB). Another popular but fundamentally different low-regret policy is Thompson Sampling [135]. Being Bayesian in spirit, it is better suited for modeling complex online problems [91], which is why we adopt this method in our recommender system. Its main idea is posterior sampling: the expected rewards of bandit arms are obtained by sampling the parameters from the posterior distribution associated with each arm. The algorithm then chooses the arm with the highest expected reward. Because of this randomized sampling procedure, the algorithm implicitly alternates between maximizing the immediate reward, and gaining new information about other arms in order to aid future rewards. This exploration-exploitation trade-off is the key aspect of the multi-arm bandit problem. The behavior of Thompson Sampling tends to be mostly exploratory in the beginning, but becomes more exploitative as it gets more and more data, allowing the algorithm to gradually converge closer to the true parameter values. We give a more formal introduction to Thompson Sampling for the Bernoulli bandit problem in Section 8.7.2.

Since most work on multi-arm bandits has been theoretical in nature [268], their implementation in real-world scenarios such as content recommendations is still elusive [3]. This is because many of the assumptions made by traditional MAB are violated in practical applications. In particular, we envision the following challenges of implementing bandit recommendations in e-commerce:

Non-stationary rewards. Since behavioral signals in the data (i.e. user-item interactions such as clicks and purchases, see Section 8.4.2) are aggregated over time, predictors of this type gradually improve their performance and are hence characterized by non-stationarities in their reward sequences. Algorithms that deal with this issue include *switching* and *restless* bandits [38, 88, 187].

Multiple plays. The "top-N ranking" problem dictates that multiple actions need to be taken in each round to ensure that all available item slots are filled. Problems of this type are addressed with *ranked* [158, 235, 274] *cascading* [156, 167] and *multiple-play* bandits [91, 172, 290].

Variable availability of actions. In the real world, not all actions are available at each time instance. For example, this may happen when predictors find no correlations between items, or when items themselves are not accessible (e.g. out of stock). Bandit policies capable to accommodate such situations are known as *sleeping* bandits [131, 142, 267].

In this paper we propose a *streaming* recommender system [296], which adapts existing MAB policies to the aforementioned scenarios. Motivated by the fact that many industrial recommenders tend to favor simple algorithmic approaches (e.g. best-seller lists) [221], our solution is essentially an ensemble learning scheme that uses Thompson Sampling to orchestrate the collection of elementary recommendation components. This allows content-based and behavioral signals in the data to effectively complement each other in response to queries. The proposed model offers a number of advantages to a prospective vendor (explained in the following pages in more detail):

- It allows to easily plug-in/out recommendation components of any type, without making changes to the main algorithm
- Modeling components as arms helps to overcome the scalability limitation of the more common "items-as-arms" representation
- Handling context can be shifted to the level of components, hence eliminating the need for contextual MAB policies
- Elementary recommendation components are easily interpretable and hence allow for more transparent user interfaces that improve user's trust in the system

8.2 Article notes

The present article constitutes an extension of the original paper (Brodén et al. [31]) published in the proceedings of the 2018 Conference on Intelligent User Interfaces. The extension offers a theoretical analysis of ensemble learning (Section 8.5) to better motivate the concept of dynamic partitioning that is central to our approach. This concept has also been explained in more detail in Section 8.6 with the help of illustrative examples.

Another notable difference from the conference version is the introduction of session-based personalization to our ensemble framework by constructing a kNN component. To this end, we provide the theory behind this component in Section 8.4.3, and perform a new experiment on session-based personalization (Section 8.8.7). Under this experiment, we also provide component statistics, showing the exposure and success rate of the most active components to better visualize the inner workings of the ensemble.

The Related work section has been expanded with a coverage of item-to-item, session-based, and sequence-aware recommender systems (Section 8.3). We also provide a more detailed explanation of the session-based recommendation mechanism illustrated in Figure 11.

Lastly, our experimental setup (Section 8.8.1) now also details the streaming evaluation protocol used in our experiments.

The paper is organized as follows. Section 8.3 gives an overview of related variants of MAB policies, along with related studies on MAB-based recommender systems. Section 8.4 introduces our approach and details the proposed ensemble algorithm in pseudo-code. In Section 8.8 we describe our experiments and present their results. We summarize and conclude our findings in Section 8.9.

8.3 Related work

The problem that we address in this paper, i.e. assisting unidentified users of the system by adaptively suggesting relevant products, is referred to as *session-based* recommendations in the literature. This is distinct from *session-aware* recommender systems, which have access to multiple user sessions of a known user. Due to the aforementioned dominance of anonymous sessions in e-commerce, session-based recommendations have attracted significant interest in this area (e.g. [47, 121, 128, 163, 286]). The conversion of anonymous sessions is especially important from the perspective of customer retention and improving future customer experience by enabling deeper personalization via user profiling. Indeed, session-based recommender systems can be thought of as weakly personalized solutions, as their knowledge of the user is limited to the scope of a single session. On the other hand, they are effective at capturing the so-called *interactional context* defined by the user's current shopping mood and intent, which can be inferred from their most recent actions [206, 233]. This is different from the more traditional *representational context* defined by less dynamic contextual factors, such as location, season, or preferences [206]. For a typical e-commerce platform featuring a large number of new or anonymous users, deriving *interactional* contextual factors is paramount [233].

The recency of actions to be considered for recommendations is defined by the user's last-N interactions on the website, where N can be as large as the length of the session, or as small as a single (i.e. most recent) event. The latter scenario ($N = 1$) effectively corresponds to *item-to-item* recommendations, where the currently examined item serves as a query to a recommender system [110]. This type of context adaptation prevails in current research on session-based recommendations [233], and is also the one we follow in this work. In Section 8.8.7, however, we also experiment with the ensemble configuration that makes use of the entire user session for generating recommendations. Considerable attention in such scenarios (i.e. when N is the total number of events in the user session) has been devoted to sequence-aware recommendation algorithms [232], particularly those based on recent advances in Recurrent Neural Nets (RNN) [47, 109, 120, 234]. In spite of being at the forefront of research in session-based recommendations, current implementations of RNN-based architectures struggle to surpass the accuracy of simpler and computationally cheaper methods such as association rules or k-nearest neighbors, as illustrated in [120, 128]. In addition, RNN-based methods suffer from poor explainability [128], which can have a detrimental effect on user experience. The above reasons justify our choice of the k-nearest neighbors algorithm for building the personalized variant of the ensemble (see Section 8.4.3).

Bandit algorithms are becoming some of the most promising approaches in the area of recommender systems [166]. Due to their exploratory nature, they are especially effective in attacking the cold-start problem (e.g. [43, 77, 181]). Referring to the e-commerce domain, Bernardi et al. [23] characterize this problem as the *continuous* cold-start, emphasizing the fact that users and items remain "cold" for an extensive period of time. Moreover, existing users have the tendency to "cool down", which happens when their previously learned intentions and preferences eventually become obsolete. This observation motivates the use of non-personalized item-to-item recommendations that we present in this work. We model bandit arms as predictors, which allows us to devise an ensemble method to tackle cold-start. This distinguishes our approach from others mentioned (e.g. [158, 164, 305, 326]), wherein arms represent single items. Whereas an item-as-arm model may be reasonable for few arms (as in [136, 164, 165, 207]), the magnitude of real-world data would likely lead to scalability issues [146].

The non-stationarity of arm rewards has been approached from different angles. In the presence of domain knowledge, the shape of the reward distribution may be known a priori. Bouneffouf and Féraud [27] show how to take advantage of this knowledge in UCB1 [14] to tackle non-stationarity in music or interface recommendations. In many cases (including ours), the functional form of non-stationarity is not known. Therefore, automatic detection of abrupt or drifting changes in reward sequences have been proposed in several extensions to UCB [88, 102] and Thompson Sampling [38, 187] policies. We will present Thompson Sampling, UCB, and some of its variants, such as UCB1, in Sections 8.7.2 and 8.8.5. The two types of behaviors (i.e. abrupt and drifting) correspond, respectively, to *switching* [88, 187] and *restless* [38] bandits. Garivier *et al.* [88] also show that any policy with a logarithmic regret for the stationary case is bound to $T/log(T)$ regret in the presence of reward irregularities, where T is the time horizon. In fact, the notion of *dynamic regret* has been introduced for drifting reward functions, which is measured w.r.t. a benchmark policy taking the best action for each round in isolation [267].

The multiple arm selection (top-N) problem is highly relevant for recommender systems. The most straightforward solution, known as *ranked bandits*, assigns a separate bandit to each slot in the ranking [235, 274]. Lacerda [158] presents an interesting use case for ranked bandits that optimizes multi-objective ranking in recommender systems. More recently, *cascading bandits* have been proposed as a variant of a ranked bandit for modeling position bias in click data [156], and have been evaluated on movie recommendations [167]. Ranked bandits have been criticized for the underutilization of the available feedback, since it cannot be shared between individual bandits [172]. A better approach in this respect is the so-called *multiple-play* bandit, which can fully observe all incoming rewards. It is this approach we adopt in our work. A multiple-play bandit algorithm called *EXP3.M*, originally proposed in [290], has been assessed against two ranked bandit algorithms on movie and joke recommendations, demonstrating that *EXP3.M* indeed converges faster than the two ranked algorithms [172]. Likewise, Gopalan et al. [91] apply Thompson Sampling for the problem of top-N arms selection and report a significant reduction in running time as compared to a decoupled bandit policy. They conclude that being Bayesian is advantageous for addressing complex bandit problems, and thus motivate our choice of policy.

Our problem setting, which resembles many other practical situations, assumes that certain actions may be unavailable or unreasonable at times. This has implications on the definition of regret, which is traditionally measured w.r.t. the best action in hindsight and is no longer applicable [142]. The idea then is to choose the best action among currently available actions ordered by their expected reward. Problem settings that exhibit such behavior have been coined *sleeping bandits* and studied in various contexts. For example, Kanade et al. [131] present a no-regret sleeping bandit algorithm for adversarial rewards, whereas Kleinberg et al. [142] analyze the regret for both adversarial and stochastic cases. A contextual sleeping bandit with stochastic rewards is presented in [267], wherein the author reproduces the result of [142] using contextual zooming. The term 'contextual' refers to bandit algorithms that explicitly take into account the state of the environment (a.k.a. the context) in a given time step to decide upon the arm selection.

Recent years have seen successful implementations of Thompson Sampling for recommendations of news articles [45, 281], movies [326], music [101], documents [28], courses [207], search queries [112], and dating partners [11]. Convincing attempts have been made to couple Thompson sampling with popular matrix factorization techniques [136, 326]. Closest in spirit to our work is the paper by Tang et al. [281], who present an ensemble meta-bandit recommender system. Similar to our approach, they use a hyper-bandit ruled by non-contextual Thompson Sampling to adaptively explore/exploit base contextual bandits that select items to recommend. The same basic idea of using a bandit algorithm for *prediction model selection* in recommender systems has been conveyed by Felício et al. [77]. They experiment with UCB1 and ϵ-greedy policies with arms modeled as clusters of users induced by matrix factorization. Our work differs mainly in how we model arms and attribute rewards. Contrary to some of the aforementioned recommenders that maximize the click-through rate (CTR) (e.g. [3, 45, 158, 164, 172, 281]), we are interested in optimizing the number of sold units. In our experience, this e-commerce objective does not benefit from items that merely attract clicks, and therefore requires a different attribution model. We outline our approach in the next section.

8.4 Approach

8.4.1 Problem setting

Following the reinforcement learning paradigm, we consider a typical e-commerce scenario as an interplay between:

- *E-commerce environment* comprised of a product catalogue $I = \{i_1, \ldots, i_{|I|}\}$, and (possibly overlapping) user browsing sessions, $S = (s_1, \ldots, s_{|S|})$, each defined as $s = (q_{t_x}, \ldots, q_{t_y}, P_{\tau_s})$, where query $q_t \in I$ represents a product page visit at time t, and $P_{\tau_s} \subseteq I$ is the final purchase order at time τ designating the end of the session (typically, $|P_{\tau_s}| \ll |I|$). Note that sessions that do not contain purchases are extremely common, but cannot be used for evaluating recommendations (however, they are useful for training prediction models).

- *Recommendation agent* implemented as a multi-arm bandit with a set of actions $\mathcal{A} = \{a_1, \ldots, a_{|\mathcal{A}|}\}$ and a sequence of states $\Sigma = (\sigma_1, \ldots, \sigma_{|\Sigma|})$. Each action represents a base recommendation function that maps a given query to some subset of the product catalog, i.e., if Q is the set of all queries, then $a : Q \to \mathcal{P}(I \setminus Q)$, where $\mathcal{P}(\cdot)$ is the powerset function.

We will only consider queries and purchases belonging to the same session and will, for ease of notation, omit session indices.

For each query q_t with an associated N-sized recommendation list L_t, the agent takes at most N available actions $A_t = (a_1, \ldots, a_{|A_t|})$, s.t. $|A_t| \leq N$ and $a_i \in \mathcal{A}$, according to some bandit policy π so that the precision at time t: $prec@N(L_t) = |L_t \cap P_\tau|/N$ is maximized. An action can appear more than once in A_t and the recommender does not have access to P_τ at time t, since $t < \tau$.

Each *display* of an action's recommendation in L_t results in a binary *reward* $r \in \{0, 1\}$ after observing a purchase order at time τ. We refer to each successful recommendation (i.e. the recommended item is eventually bought) corresponding to $r = 1$ as a *hit*. The reward of an action that fails to generate a hit is zero ($r = 0$).

In case the same successful recommendation is given by different actions within a session, we face the problem of *attribution*, i.e. deciding which action should get credit for the sale of an item to the active user [17]. We follow the *equal* multi-touch attribution strategy, according to which each action that generates a hit is equally rewarded. Alternative strategies include first-touch attribution, last-touch attribution, and fractional attribution (see [17] for details). Since we have no means of establishing which of the displayed recommendations are actually noticed by the user, we consider equal attribution to be the safest choice.

For each action $a \in \mathcal{A}$, we maintain a tuple of running rewards r_a and displays n_a, which represents the agent's current state of knowledge: $\sigma_j = (r_a, n_a)_{a \in \mathcal{A}}$. The agent moves to an updated state σ_{j+1} after attributing rewards at time τ. Figure 11 exemplifies the described mechanism on the example of a record store. As mentioned earlier, each user query corresponds to a particular product page visit, where the recommendations are to be shown. In other words, even if a user performed a search query, the term 'query' in our case refers to a click on one of the search results, which triggers recommendations for a particular item (or the user-item pair, if the user is identifiable). As Figure 11 shows, the agent receives a query q_1 from a user who accesses a particular record page at time t_1. Based on its state σ_1 and policy π, the agent chooses three actions, namely a_1, a_2 and a_3, each returning a single item as a recommendation. This results in three recommendations being shown to the user on the product page of the query item q_1. The user's browsing session continues at step t_2 with another product page visit and corresponding recommendations from the agent. At time τ, the user buys three items, all of which were recommended at some point during the session. This forms the basis for attributing rewards to bandit arms (or 'actions'). These attributions are shown in dotted lines in Figure 11. After the attribution round, the agent moves to a new state σ_2 with a newly acquired knowledge. In reality, the agent operates on multiple sessions similar to the one depicted in Figure 11, which may be active at the current time step. The ultimate goal of the agent is to maximize the cumulative reward: $r_T = \sum_{t \leq T} r_t$, where r_t is the sum of rewards obtained at time t.

The described scenario showcases the interactive aspect of the system: as the agent influences the user's behavior by giving recommendations on product pages, the user's reaction to these recommendations in turn helps the agent adapt and improve its action selection policy. What Figure 11 also demonstrates is the transparency of this user-agent interaction, which is achieved by designing explainable actions corresponding to a specific item attribute (e.g. 'artist') or event (e.g. 'viewed after item x'). Recommendation interfaces displaying such explanations next to each recommended item can serve as an additional navigation tool allowing users to explore relevant items based on some specific attribute of interest (for example, browsing books written by the same author). Next, we formally introduce the two types of actions mentioned above.

8.4.2 Base recommenders

As stated earlier, action set $\mathcal{A}$ is an ensemble of base recommendation functions or "components".[24] These are simple contextual predictors, whose output is determined by the input query. Each component defines a graph $G_a((V, E), W)$, where V is a set of vertices representing items, E is a set of edges representing between-item associations (in-

[24]The terms 'actions', 'components', 'predictors', and 'recommendation functions' have the same meaning and are used interchangeably

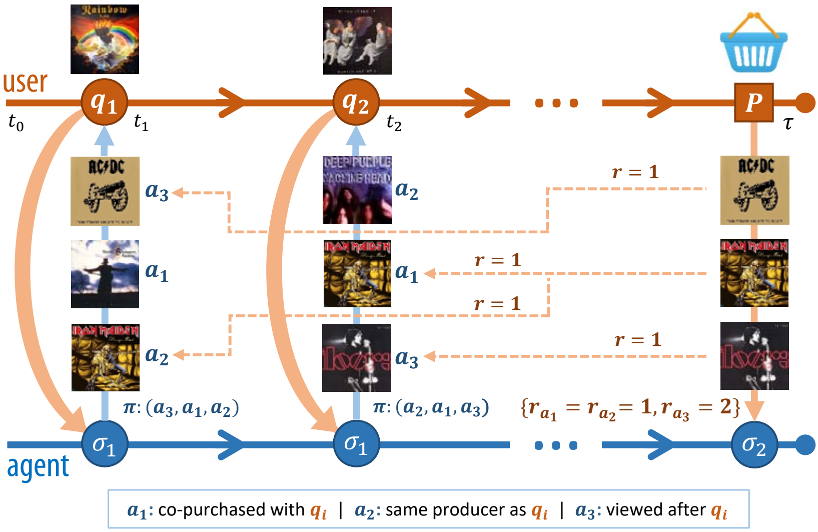


Figure 11: *i2i* session-based recommendations with explainable actions

duced by a), and W is a set of edge weights measuring the strength of these associations. We assume that all edge weights are positive.

We distinguish between two types of components:

1. **Attribute-based** components connect items via their content features (i.e. metadata). We use elementary components, each representing a particular attribute of the product catalog. Edges of this type are undirected. For example, $N_{G_{color}}(v)$ retrieves the neighborhood of item v containing all items of the same color(s) as v. Optionally, edge weights can be used to measure the number of attribute values that are matched.

2. **Behavioral** components connect items via events (clicks, purchases, etc.). The edge's weight in this case captures the frequency of event observations, i.e. the strength of the association. These values will be used as estimations of expected reward of this recommendation response. The edges can be directed, indicating that the event connecting two items is only meaningful in one direction. For example, we use the following four types of behavioral components:

- ***click-click*** *(cc)*: the directed edge $(v, v') \in E[N_{G_{cc}}(v)]$ signifies that the click on item v is followed by the click on item v'.
- ***session-click*** *(sc)*: the edge $\{v, v'\} \in E[N_{G_{sc}}(v)]$ signifies that items v and v' are both clicked within the same session.
- ***purchase-purchase*** *(pp)*: the directed edge $(v, v') \in E[N_{G_{pp}}(v)]$ signifies that the purchase of item v is followed by the purchase of item v'.
- ***customer-purchase*** *(cp)*: the edge $\{v, v'\} \in E[N_{G_{cp}}(v)]$ signifies that items v and v' are both purchased by the same customer (not necessarily in the same session).

The two types of components correspond, respectively, to the terms *content-based* and *collaborative* filtering used in the recommender systems literature. Our choice of very granular predictors is dictated by the need for cross-domain applicability, since we want the recommender to operate well on various types of e-commerce data from complete cold-start. However, a vendor can engineer components in many ways, for example by introducing more complex feature combinations, campaigns, best-seller/special offer lists, or personalization. Next, we discuss the construction of a session-based personalized component using a neighborhood-based approach.

8.4.3 k-Nearest Neighbors (kNN) component

The introduction of a personalized component to the ensemble enables it to look beyond the current query and take advantage of the entire user session for the computation of recommendations. This has a potential to boost the overall accuracy of the ensemble by leveraging the power of collaborative filtering to serve session-tailored content.

Neighborhood-based methods such as kNN are among the most widely practised personalized recommendation techniques due to their effectiveness and ease of implementation. As we shall see in Section 8.8.7, the addition of this component to the ensemble will have a considerable positive impact on the accuracy of recommendations, which is why we take a closer look at this method here. According to recent empirical evidence, the predictive performance of kNN on session data can surpass that of the state-of-the-art deep learning methods such as `GRU4REC` [120]. In session data, the neighbors in kNN refer to sessions that are in some sense similar to the active session.

Furthermore, kNN algorithms are incremental by their nature, making them a good choice for streaming session data. This means that kNN is naturally capable of handling new sessions having only one event (in which case it becomes equivalent to 'click-click' or 'session-click' component), or recommending new items as long as at least one session containing the new item may act as a 'neighbor' to some other session. Another important property of kNN is its ability to provide explainable recommendations, which is difficult with other recommendation techniques based on e.g. Singular Value Decomposition or Recurrent Neural Networks. Explanations can be given on the grounds of shared items between the current session and its neighbors from which the recommendations are extracted. As mentioned earlier, the explainability of recommendations is a very important factor of the user-system interaction.

The basic kNN formula for calculating the relevance score $rel(i)$ of some candidate item i for session S is:

$$rel(i|i \notin S) = \sum_{S' \in \mathcal{N}(S)} w(i, S') \cdot sim(S, S') \cdot \mathbb{1}(i \in S') \tag{29}$$

where $\mathcal{N}(S)$ is the k-sized neighborhood of session S and $w(i, S')$ is the worth of item i in session S' weighted by some similarity measure $sim(S, S')$. Typically, Cosine or Jaccard similarity is employed depending on whether the session is represented as a real-valued or binary vector. The parameter k has to be established empirically from the data.

Session representation We assume that e-commerce sessions are comprised of three types of events: *clicks*, *additions to cart*, and *purchases*. This assumption is based on examining the data format used in some available public e-commerce datasets containing session events[25]. We note that wish-listing, tagging, or reviewing are also examples of valid event types that are, however, less common in this domain. Taking the worth of these events into account, the following holds: $w(\textit{click}) < w(\textit{cart}) < w(\textit{purchase})$. If we set $w(\textit{purchase}) = 1$, other event types can be expressed as fractions of a purchase, e.g. $w(\textit{click}) = 0.2$ and $w(\textit{cart}) = 0.5$. With this weighting scheme, we represent each session as a vector of cumulative item weights: $S = (w(i_1), w(i_2), \ldots, w(i_{|S|}))$. For example, if item i was clicked 2 times, then added to cart, but not purchased, then its worth in the session is $w(i) = 2 \cdot 0.2 + 0.5 = 0.9$. We note that it is possible to use different representations of the same session in the similarity computation phase and in the recommendation phase. For instance, we could set $w_{sim}(\textit{click}) = 0.2$, $w_{sim}(\textit{cart}) = 0.5$ to compute the neighborhood, but $w_{rec}(\textit{click}) = 0.0$, $w_{rec}(\textit{cart}) = 0.0$ to enforce recommendations of only previously purchased items. This way, higher flexibility in modeling the kNN component is achieved.

Ageing The worth of an item in the session may also depend on how many time steps back it was observed. This captures the intuition that the most recently examined items are better estimators of the user's intent in the session as compared to more distant observations. Likewise, the process of establishing close neighbors may also be sensitive to their age. These two dimensions of ageing can be modelled using the concept of *half-life*, which can be defined as the time it takes for exactly half of the item or session weight to decay. The exponential decay of initial weight w_0 at time step t using the pre-set half-life parameter $t_{1/2}$ is given as follows:

$$w_t = w_0 \left(\frac{1}{2}\right)^{t/t_{1/2}} \tag{30}$$

We note that t in our case does not represent timestamp, but the temporal distance of the item/session from the most recent item/session. In this way, session similarities and item weights in Equation 29 are time-decayed using Equation 30.

[25] **http://2015.recsyschallenge.com/challenge.html**, **https://www.kaggle.com/retailrocket/ecommerce-dataset/**

Algorithm The kNN algorithm scales poorly if all known sessions are considered as candidates for neighborhood formation. It is therefore common to perform a pre-processing step that selects some manageable subset of sessions as possible good candidates for the neigborhood. At first, this can be achieved by retaining all sessions in which at least one item of the current session appears. This enables different strategies for pre-preprocessing using efficient lookup and set operations. In our experiments, the strategy that yields the best results simply considers sessions containing the current query item. If this set is empty, then the union of sessions containing previous items of the current session is returned. For faster performance, it is also possible to return top-k sessions sorted by their overlap with the current session. To demonstrate the effectiveness of pre-processing, we report running times and accuracy metrics of kNN *with* and *without* pre-processing in Experiment 5. Once the possible candidate sessions have been found, kNN proceeds as shown in Algorithm 3.

Algorithm 3 kNN recommender

Input: $\mathcal{D}$: dataset, S: current session vector, k: number of neighbors
Output: recommended item i^*

$\mathcal{N}(S) \leftarrow \textit{pre-process}(\mathcal{D})$
for each $S' \in \mathcal{N}(S)$ **do**
  $\textit{sim}(S, S') \leftarrow \frac{S \cdot S'}{||S|| \cdot ||S'||}$ ▷ cosine similarity
if $|\mathcal{N}(S)| > k$ **then**
  $\mathcal{N}(S) \leftarrow \arg\max_{\mathcal{N}'(S) \subset \mathcal{N}(S), |\mathcal{N}'(S)| = k} \sum_{S' \in \mathcal{N}'(S)} \textit{sim}(S, S')$
$\textit{rel}(i) \leftarrow 0.0, \forall i \in S' \in \mathcal{N}(S)$
for each $S' \in \mathcal{N}(S)$ **do**
  for each $i \in S', i \notin S$ **do**
    $\textit{rel}(i) \leftarrow \textit{rel}(i) + \textit{sim}(S, S') \cdot w_{\textit{rec}}(i, S')$
$i^* \leftarrow \arg\max_i \textit{rel}(i)$
return i^*

8.5 Analysis of ensemble learning

To further illustrate the issues with ensemble learning, let us for now assume that we are in possession of an optimal algorithm that has full knowledge of the incurring expected rewards associated to each query in the input sequence, i.e., $\mathbb{E}(R_t(L_t(q_t))) = r_{L_t,t}$, is known to the algorithm, given that $R_t(L_t(q_t))$ denotes the random variable of rewards for the recommendations $L_t(q_t)$ associated to the query t. The notation $r_{L_t,t}$ is short-hand for this expected reward. The total expected reward for the algorithm is $\max_{L_1^*,\ldots,L_T^*}\{\sum_{1\leq t\leq T} r_{L_t^*,t}\}$, where T is the time horizon. The optimal algorithm simply picks, at each time step t, the recommendations L_t^* with maximum expected reward, $r_{L_t^*,t} \geq r_{L_t,t}$, for all L_t.

An ensemble learner that is supposed to learn the optimal algorithm faces several problems. First, non-stationary rewards are impossible to learn unless there is some inherent structure or some known restrictions to them. Second, and possibly even worse, is that in e-commerce settings, the number of arms, i.e., the size $|I|$ can be in the millions and the number of permutations of recommendation items thus many magnitudes larger. The expected reward for each arm is therefore extremely small, thus the speed of convergence for any bandit based ensemble learner will be very low.

To address the convergence speed, we would like to reduce the number of arms and at the same time increase the expected rewards of these. Going back to our fictional optimal algorithm, we assume that it has additional access to base recommendation functions or components $a_1, \ldots, a_n$, which we further assume always produce locally optimal recommendations. Thus, our optimal algorithm has full knowledge of the expected rewards $\mathbb{E}(R_t(a_j(q_t))) = r_{j,t}$, where $R_t(a_j(q_t))$ denotes the random variable of rewards for the recommendations associated to component a_j on query q_t at time t. This leaves the problem of non-stationary rewards unsolved, but for now let us make the further assumption that $\mathbb{E}(R_t(a_j(q_t))) = r_j$, i.e., the reward is stationary over time. We can then consider the reward r_j to be associated directly to the component a_j. Again, an optimal algorithm in this case simply picks the component that has the largest reward r_j and uses this one throughout the query sequence. The objective of an ensemble learner is hence to learn which component has the maximum expected reward and once this is done, it continues answering the remaining queries using this component, since the reward probabilities are stationary. In this setting, Thompson sampling has been proved to have low regret [135] and with few components, the convergence speed increases accordingly.

For some practical applications, the assumption that each component a_j has a stationary reward is valid, e.g., in the comparison of the efficiency of medical treatments as originally proposed by Thompson [284]. However, for most e-commerce situations, this assumption is completely unreasonable leading to poor performance of ensemble learners like Thompson sampling and UCB. In fact, an ensemble learner will learn the average reward of the component a_j with the best average performance over the query sequence. This means that the ensemble learner will never do better than the component with best average performance even if there exist other components that do significantly better when queries are in specific domains or contexts.

Algorithm 4 Optimal Full Knowledge Recommender

Input: q_t: a query at time step t, a_j, for $1 \leq j \leq n$: components

Output: Optimal top-N recommendations L_t, for time step t

```
1: j* ← 0,  r* ← 0.0
2: for each 1 ≤ j ≤ n do
3:     if q_t ∈ a_j and r_j > r* then                ▷ Assume true for some j
4:         j* ← j,  r* ← r_j
5: return L_t(a_j*(q_t))                ▷ Component a_j* has maximum reward for q_t
```

To overcome this problem, we relax our strong assumption about stationary rewards and make a significantly weaker assumption on the reward probabilities. Each component a_j can have different probabilities for different queries, but the queries can be partitioned into two sets Q_j and $\bar{Q}_j$, such that queries $q_t \in Q_j$ each have the same expected reward r_j and the queries $q_t \in \bar{Q}_j$ each have expected reward

$\mathbb{E}(R_t(a_j(q_t))) \ll r_j$, i.e., much smaller than r_j. We henceforth identify each component a_j with the corresponding query set Q_j. In the sequel, some components (the behavioral ones) are given by enumerating the queries and some (the attribute-based ones) are defined intensionally, by reference to the corresponding attributes, e.g. jazz records.

Consider now the algorithm presented as Algorithm 4. We prove that Algorithm 4 is indeed optimal.

Theorem 1. *Given n components $a_1, \dots, a_n$ with associated rewards $r_1, \dots, r_n$ and a query sequence $Q_T = \{q_1, \dots, q_T\}$, such that each $a_j \subseteq Q_T$, then Algorithm 4 produces recommendations with maximum total expected reward.*

Proof. Let $J_{\mathrm{OPT}} = \{j_1^*, \dots, j_T^*\}$ be the sequence of indices of components giving the maximum expected reward. The total expected reward for this solution is $\sum_{1 \le t \le T} r_{j_t^*}$. Consider a query q_t at time step t where Algorithm 4 does not use index j_t^* but some other index j_t^+ in Step 5 of the algorithm. Since Algorithm 4 always selects a component having maximum associated reward, this means that $r_{j_t^+} \ge r_{j_t^*}$ and the choice of Algorithm 4 is at least as good as the optimal choice. Repeating the argument for every t, $1 \le t \le T$, proves the theorem. $\square$

For this, less restrictive, model, we have two challenges. First, we need to modify the ensemble learner to be able to acquire sufficient knowledge to apply Algorithm 4 rather than simply the component with best average performance. The main difference lies in the test $q_t \in a_j$ in Step 3 of Algorithm 4. Thus, for the ensemble learner to work, it must be able to perform this test on each component, i.e., test that the query is contained in the component and hence that the recommendations produced have the high reward r_j associated to the component a_j. In Section 8.7, we develop such an ensemble learner; see Algorithm 6. However, instead of implementing the test in the learner, it defers the test to the component which then decides to answer if the query belongs to it, otherwise it refuses to provide recommendations for the query.

The second challenge is to guarantee stationary or almost stationary rewards for the queries in each component (as opposed to stationary rewards for all queries). Since the ensemble learner has no a priori knowledge of the rewards, it may falsely include or exclude queries from a component. As the actual rewards are learned, the ensemble learner must update the components to contain the correct queries so as to maintain almost stationary component rewards.

To ensure that each component maintains nearly stationary rewards on its queries throughout the query sequence, we introduce the concept of *dynamic partitioning* of components to let the ensemble learner split components into pieces when rewards of the queries contained in the components begin to differ too much. Again, this partitioning is deferred to the actual component so as to avoid introducing domain specific actions to the ensemble learner. We discuss the dynamic partition process in more detail in the next section. The expected rewards of components or predictors will henceforth be referred to as the *precision* of these.

8.6 Dynamic partitioning

It is apparent that bandit arms constructed as in Section 8.4.2 are prone to non-stationary behavior, since the size of the neighborhood $N_{G_a}(v)$ may differ significantly from query to query. The problem is particularly profound in behavioral components, as new edges are constantly added to the graph and the weights of existing edges are constantly updated from new observations. This may lead to situations when top-performing predictors (in the long run) get under-explored by the bandit algorithm because of their poor performance at an early stage.

A simple solution to this problem, as we indicated in the previous section, is to dynamically partition a component a into several disjoint parts or "sub-components" based on the estimated precision of each candidate item contained in the query response $N_{G_a}(q)$. In other words, we cluster the queries with the aim to obtain sub-components each having a distinct but homogeneous precision. Homogeneous here means that within each sub-component, the precision is roughly the same. We achieve this by actually partitioning the query-response pair to simultaneously partition the component and to cluster the responses according to their precision estimates. This approach is consistent with the *mixture of experts* paradigm, which "models the local competence of experts in different distributions of data space according to each input data" [182]. In our case, each sub-component can be viewed as an expert that specializes in certain types of input queries. Let

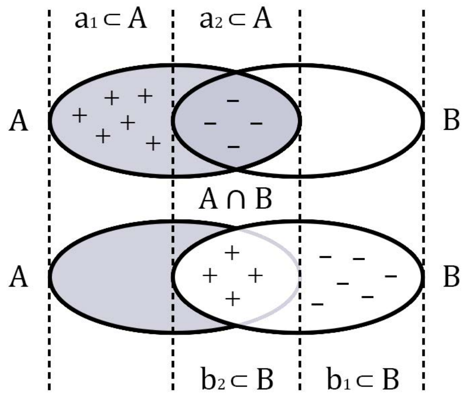


Figure 12: Partitioning the components. Top: query answers of component A. Bottom: query answers of component B. The area $A \cap B$ contains common queries for A and B (see further explanation in text).

A_a denote a *partitioning* of a recommendation component a: $A_a = \mathcal{P}(a), s.t. \bigcap_{a' \in A_a} = \varnothing, \bigcup_{a' \in A_a} = a, \forall a \in \mathcal{A}$. We can now expand the original action set to the set of sub-actions: $\mathcal{A} = \bigcup_a A_a$, where each sub-action $a' \in A_a$ has a (relatively) fixed reward distribution. The rationale for partitioning the components is illustrated in Figure 12. In this example, components A and B answer 10 queries each, with 4 queries in common. The successful and unsuccessful answers are marked with "+" and "-", respectively. Overall, A is more precise than B since it correctly answers 60% of the queries vs. 40% for component B. However, B happens to be much better in segment $A \cap B$, with 100% precision. The total precision for all queries is measured as $prec_{total} = prec(A - B) + prec(A \cap B) + prec(B - A)$. If A and B are used 'as is', the precision comes up to $prec_{unpartitioned} = 1 \cdot 6/10 + 0.4 \cdot 4/10 + 1 \cdot 0/10 = 76\%$. However, if A is partitioned into segments a_1 and a_2 and B is partitioned into segments b_1 and b_2 (with an overlap in a_2 and b_2) as shown in Figure 12, the precision becomes: $prec_{partitioned} = 1 \cdot 6/10 + 1 \cdot 4/10 + 1 \cdot 0/10 = 100\%$. This is happening because for segment $A \cap B$, where sub-component b_2 has perfect precision, it will be preferred to a_2 at all times by the sampler. The overall gain of this partitioning is thus 31.6%. It can be shown that the gain by a factor of 2 is achievable in the best-case scenario.

Since the actual precisions are not known a priori, we have to resort to various proxies for precision depending on the type of component. For example, attribute-based components can be partitioned based on the *inverse document frequency* (IDF)[26], as shown in Algorithm 5.

Algorithm 5 IDF-based partitioning

Input: q: query, α: attribute, I: item catalog, K: #subsets in partition
Output: A_α: partition of candidate itemset $V(N_{G_\alpha}(q))$

```
α'_1 ← ∅, α'_2 ← ∅, ..., α'_K ← ∅          ▷ initialize sub-components to empty sets
A_α ← (α'_1, α'_2, ..., α'_K)              ▷ create a partition
idf_1 ← 0, idf_2 ← 0, ..., idf_|I| ← 0     ▷ initialize IDF scores of each item to 0
for each v ∈ attribute_values(α, q) do
    C ← V(N_{G_{α:v}}(q))                  ▷ generate candidate itemset induced by value v
    idf ← log(|I|/(|C| + 1))               ▷ calculate the inverse document frequency
    for each i ∈ C do                      ▷ for each item in candidate itemset
        idf_i ← idf_i + idf                ▷ update the IDF value of an item
for each i ∈ V(N_{G_α}(q)) do              ▷ for each item in α's response to q
    k ← select_index(i, idf_i)             ▷ get sub-component from item's idf
    α'_k ← α'_k ∪ {i}                      ▷ assign the item to the selected sub-component
return A_α
```

The intuition behind it is that the more frequent the target attribute value is in other items, the more difficult it is to make the right recommendation. This is captured by the IDF score:

$$IDF(q, a) = \log \frac{|I|}{|N_{G_a}(q)| + 1} \tag{31}$$

[26]Term frequency (TF) is omitted from calculations as the number of occurrences of an attribute value in the same item is rarely greater than 1

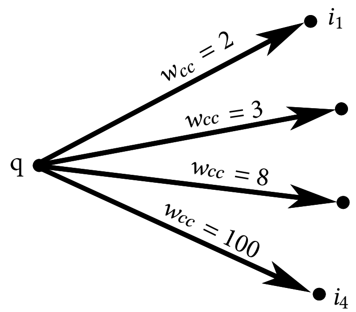
where $|N_{G_a}(q)|$ is the number of retrieved items sharing the same value of attribute a with the query item q, and $|I|$ is the size of the item catalog. Thus, smaller neighborhoods $N_{G_a}(q)$ result in better (higher) IDF scores. These scores are aggregated when the query item has multiple values for the attribute (e.g. a book with several authors), as Algorithm 5 shows.

When it comes to behavioral components, we can directly use edge weights $w_a(v, v')$ corresponding to event observations as a proxy for precision (for the kNN component, we use the relevance score computed using Equation 29). The obtained scores are then assigned to bins to obtain the partitioning of the component's output using, e.g., $\log_2(score)$ for bin boundaries. Because the event graphs are constantly evolving, partitioning is done dynamically upon each response $N_{G_a}(q)$ of some component a to some query q, and hence new (more precise) sub-components emerge with time. For example, as soon as an association between any pair of items surpasses the previously unconquered threshold of φ observations, it will automatically be assigned to a newly created partition corresponding to $w_a(v, v') \geq \varphi$. In this manner, pairwise item associations accumulating more weight eventually migrate from a lower-precision partition to a higher-precision one. This ensures that the precision of each partition is kept within certain limits, thus maintaining almost stationary reward estimates. Figure 13 gives a toy example of partitioning a response of a 'click-click' component to a query on item q. We see that this response yields four recommendable items (nodes i_1 to i_4). For each association between a query item and a recommendable item, we calculate the precision estimate by taking the logarithm of the associated weight (recall that it corresponds to the event observation count). The obtained estimates determine how to partition the response by clustering the four edges - (q, i_1) to (q, i_4) - to sub-components, based on some pre-set thresholds for bin boundaries (in our example, 2 and 6). Note that as soon as the edge (q, i_2) receives one more observation resulting in $w_{cc}(q, i_2) = 4$, this edge will migrate from sub-component a_1 to a_2.

Given: query q, component A = *'click-click'* (cc), thresholds $\Phi = \{2, 6\}$

$N_{G_{cc}}(q) = \{i_1, i_2, i_3, i_4\}$

$\widehat{prec}(q, i_x) \overset{\text{def}}{=} \log_2 w_{cc}(q, i_x)$



$\widehat{prec}(q, i_1) \overset{\text{def}}{=} \log_2 w_{cc}(q, i_1) = 1$

$\widehat{prec}(q, i_2) \overset{\text{def}}{=} \log_2 w_{cc}(q, i_2) = 1.58$

$\widehat{prec}(q, i_3) \overset{\text{def}}{=} \log_2 w_{cc}(q, i_3) = 3$

$\widehat{prec}(q, i_4) \overset{\text{def}}{=} \log_2 w_{cc}(q, i_4) = 6.64$

$a_1 : 0 \le \widehat{prec}(q, i_x) < 2 \Rightarrow a_1 \ni (q, i_1), (q, i_2)$

$a_2 : 2 \le \widehat{prec}(q, i_x) < 6 \Rightarrow a_1 \ni (q, i_3)$

$a_3 : 6 \le \widehat{prec}(q, i_x) \Rightarrow a_1 \ni (q, i_4)$

Figure 13: An example of dynamic partitioning of a behavioral component ('click-click') into three sub-components a_1, a_2 and a_3, in response to query q.

8.7 An ensemble learning agent

8.7.1 Preliminaries

Our agent extends beyond the classical multi-arm bandit, since it needs to solve the top-N selection problem with varied action availability. The availability of action a for filling the j-th position in the top-N list L_t is subject to the following conditions:

1. $N_{G_a}(q_t) \neq \varnothing$, i.e. the response of action a to query q_t is non-empty

2. $N_{G_a}(q_t) \backslash L_t[1, \dots, j-1] \neq \varnothing$, i.e. the response contains at least one item distinct from those already recommended at the first $j-1$ positions of the recommendation list L_t.

Based on the above conditions and the policy π, the agent plays the best among currently available actions. This corresponds to picking one item from the action's sub-graph. The procedure is repeated for each position in L_t. The ensemble learner is therefore characterized as a *sleeping* bandit with *multiple plays*. Both recommendation steps (i.e. choosing an action and choosing an item) can be taken in either deterministic or probabilistic manner.

Because of the partitioning method presented earlier, we can apply any standard stochastic policy to our bandit ensemble framework, henceforth referred to as BEER (Bandit Ensemble for E-commerce Recommendations), where action rewards are assumed to be drawn i.i.d. from some fixed probability distributions. This also means that the selected action can now pick a *random* item from its sub-graph, which helps to diversify recommendations without loss in precision.

A well-studied family of *deterministic* MAB policies is based on the principle of *optimism in the face of uncertainty*. These policies select an action with the highest index corresponding to the *upper confidence bound* (UCB) for the action's expected reward, plus some padding function (e.g. [88, 159, 164]). They capture the intuition that the high bound is either due to the action being under-explored, or well-rewarded in expectation. Other policies balance between exploration and exploitation *probabilistically*. For example, the ϵ-*greedy* policy chooses a random action with probability ϵ and the empirically optimal action with probability $1 - \epsilon$, whereas *randomized probability matching* chooses an action in proportion to its probability of being optimal. The latter approach is known as Thompson Sampling and has shown promising results in many practical applications, including top-N action selection [91]. We now give an overview of this policy and then explain how to adapt it for the needs of our ensemble learning bandit (Algorithm 6).

8.7.2 Thompson sampling

Probability matching is a Bayesian heuristic, which models reward distribution using a parametric likelihood function $P(r|a, \theta)$, where $\theta = (\theta_1, \ldots, \theta_{|\mathcal{A}|})$ is some unknown parameter vector. Our reward system implies a Bernoulli bandit, where parameters θ represents the probabilities of obtaining a binary reward. Given agent state σ at the current step t' and some prior distribution over θ, the posterior distribution is obtained using Bayes' rule:

$$P(\theta|\sigma) = \frac{P(\sigma|\theta)P(\theta)}{P(\sigma)} \propto \prod_{t=1}^{t'} P(r_t|a \in A_t, \theta)P(\theta) \tag{32}$$

We then take an optimal action according to the probability matching principle:

$$a^* = \arg\max_a \int \mathbb{1}\left[\mathbb{E}(r|a,\theta) = \max_{a'} \mathbb{E}(r|a',\theta)\right] P(\theta|\sigma)d\theta \tag{33}$$

Instead of computing the integral, Thompson Sampling draws random samples independently for each action a from the respective posterior $P(\theta_a|\sigma)$ and chooses the action a^* with the largest θ_a. The prior is typically Beta-distributed, since it is conjugate to the Bernoulli distribution: $P(\theta_a) \sim$ Beta(α, β), where $\alpha = \alpha_0 + r_a$ can represent the number of "successes", and $\beta = \beta_0 + n_a - r_a$ can represent the number of "failures".

The randomized nature of Thompson Sampling implies continuous exploration in situations when deterministic UCB policies get stuck in the same action over the extended period of time. This makes Thompson Sampling more robust in various practical scenarios, namely:

- *Delayed feedback*. Because of random sampling, the exploration phase does not suffer from the lack of regular posterior updates.
- *Adding new actions*. One can simply set the new action's prior to the default $Beta(1,1)$ and let the ensemble algorithm continue its exploration naturally. In UCB, the exploration would focus exclusively on the new action until its confidence bound becomes comparable to the bounds of other actions.
- *Non-stationary actions*. When a previously well-performing action starts failing, Thompson Sampling goes to the exploration phase faster than UCB.

All the above scenarios are handled seamlessly without any algorithm parameterization, which is another advantage of Thompson Sampling. However, the existence of the Beta parameters allows to re-shape the reward distributions beforehand, as we discuss next.

8.7.3 Sampler priming

The parameters $\alpha > 0$ and $\beta > 0$ control the shape of Beta distribution with mean $\mu = \frac{\alpha}{\alpha+\beta}$ and variance $\sigma^2 = \frac{\alpha\beta}{(\alpha+\beta+1)(\alpha+\beta)^2}$. The prior parameters α_0 and β_0 allow us to prime the sampler with some initial frequencies of successes and failures. At time t_0, $\alpha = \alpha_0$ and $\beta = \beta_0$. The de facto choice $\alpha_0 = \beta_0 = 1$ corresponds to the uniform distribution (i.e. uninformative prior). We consider two practical situations when more informative prior parameters can be specified:

1. Recorded event data is available beforehand. This can happen when the recommender system is deployed on a pre-existing e-commerce site. In this case, parameters α and β can be learned directly by running Thompson Sampling through the recorded data. We experiment with this scenario in Section 8.8.6.

2. No recorded event data is available. This is the case of newly launched e-commerce sites, where only the product catalog is known. In this scenario, a vendor might attempt to prime attribute-based components based on the statistical properties of the product catalog. We give an example of such priming (along with its experimental results) in Section 8.8.6.

In cases when the direct estimation of α and β does not seem feasible (as in case 2), we might still make assumptions about the means of prior distributions based on some domain knowledge (e.g. product catalog). We can then reason about the variance through the following formula:

$$\sigma^2 \stackrel{\text{def}}{=} \frac{\mu(1-\mu)}{\nu}, \tag{34}$$

where $\nu > 1$ is a parameter expressing our uncertainty about the estimated mean. This parameter is defined as $\nu \stackrel{\text{def}}{=} \alpha+\beta+1$. Choosing high values of ν (i.e. high certainty) results in more peaky curves, whereas lower values lead to flatter distributions. Knowing the estimates of μ and σ^2 allows us to compute α and β as follows:

$$\alpha = -\frac{\mu(\sigma^2+\mu^2-\mu)}{\sigma^2} \qquad \beta = \frac{(\mu-1)(\sigma^2+\mu^2-\mu)}{\sigma^2} \tag{35}$$

8.7.4 Algorithm

Our bandit ensemble based on Thompson Sampling ("BEER[TS]") is summarized in the pseudo-code of Algorithm 6.

Algorithm 6 BEER[TS]

Input: $\alpha_0^{(a)}, \beta_0^{(a)}$: prior parameters $\forall a \in \mathcal{A}$ ▷ default: $\alpha_0 = \beta_0 = 1$
Output: top-N recommendations L_t, where t is current time step

- **for each** t **do**
  - **for** $j = 1, ..., N$ **do**
    - **for each** $a \in \mathcal{A}$ **do**
      - $C_a \leftarrow V(N_{G_a}(q)) \backslash L_t[1, \ldots, j-1]$ ▷ a's answer to q
      - **if** $C_a = \varnothing$ **then**
        - continue ▷ do not take "sleeping" actions
      - $A_a \leftarrow$ partition(C_a) ▷ into disjoint subsets
      - **for each** $a' \in A_a$ **do**
        - Select index $k(a')$ ▷ sub-action to sample from
        - $\theta_{a'} \sim \text{Beta}(\alpha_0^{(a)} + r_{k(a')}, \beta_0^{(a)} + n_{k(a')} - r_{k(a')})$
    - $a^* \leftarrow \arg\max_{a'} \theta_{a'}$
    - $L_t[j] \leftarrow$ pick_item(C_{a^*})
    - Observe reward $r \in \{0, 1\}$ for $L_t[j]$ ▷ delayed until time τ
    - $r_{k(a^*)} \leftarrow r_{k(a^*)} + r$ ▷ update action rewards
    - $n_{k(a^*)} \leftarrow n_{k(a^*)} + 1$ ▷ update action displays
  - Output L_t ▷ output recommendations at time t

We note that it is easy to replace TS with other MAB policies within the BEER framework. The experimental evaluation of different policies is given in Section 8.8.5.

One step of Thompson Sampler takes constant time, and the time complexity increases linearly in the number of sub-components comprising the learner.

8.8 Experiments

8.8.1 Datasets and experimental setup

We conduct our experiments using two proprietary e-commerce datasets[27] from the *books* and *fashion* domains, and one public dataset[28] [21] with *cross-domain* merchandise. From these datasets, we select 500,000 events of timestamped session data for our experiments. To make the evaluation possible, we only consider sessions that culminate in a purchase order (however, non-converted sessions can still be used to train the model). The summary of the datasets is given in Table 18.

Table 18: Dataset summary

Dataset	**Books**	**Fashion**	**Yoochoose**
# Products	91433	19731	52739
# Purchases	134850	50096	101763
# Attributes	45	51	2
# Sessions	58465	21411	55600
Mean session length	8.6 events	22.37 events	9.25 events

The composition of the ensemble that is used in our experiments varies across the datasets. While we use a fixed set of four behavioral predictors (those presented in Section 8.4.2), the number of attribute-based components ranges from 2 to 51, as Table 18 shows.

Our simulation adopts the *streaming* evaluation protocol, which has started gaining attention of the recommender systems community in recent years (e.g. [44, 170, 296]). This approach aims to simulate real-time user interaction with the system in offline mode by processing one event at a time. This is different from the traditional *batch* evaluation method, where the algorithm is first trained on a chunk of training data, and then evaluated on a disjoint chunk of test instances. In contrast, the streaming method operates on a continuous flow of data and thus does not require data pre-processing. This also means that

[27] Provided by Apptus Technologies: `http://www.apptus.com`
[28] Provided by Yoochoose: `http://2015.recsyschallenge.com`

training must be done incrementally after each observed user-item interaction. For each new event in the stream, our simulation proceeds as follows:

1. if it is a click, generate a recommendation list for the given item based on the information known so far; then go to step 3.
2. if it is a purchase, evaluate all previous recommendations given in the current session against the purchased items.
3. update the model with the observed event.

It is easy to see that this protocol is consistent with Figure 11, which was used earlier to illustrate the workings of the algorithm in a user session. Next, we present the evaluation metrics that are used in step 2 of the above protocol.

8.8.2 Evaluation metrics

In all our experiments, top-N recommendations are evaluated at $N = 5$. The accuracy of such recommendations is traditionally measured in terms of *precision* and *recall*. Precision shows how accurate the recommender system is in relation to the number of displays within a session, whereas recall evaluates its accuracy with respect to the number of purchases in a session. In the context of e-commerce, recall is an especially valuable metric since it basically shows how many purchases might have been triggered by recommendations (we note that a recommendation panel is clearly neither the only nor necessarily the main way to generate a purchase on a website). Apart from the apparent business value, converting a session via recommendations — as estimated by recall — also means that the recommender system has successfully assisted the user in finding the product(s) of their interest, thus improving the overall user experience.

Let h_s and h'_s denote, respectively, the number of total and distinct hits in session s. Note that a hit refers to a successful recommendation (which may not necessarily be unique within a session, as the same item could appear in several recommendation lists). The above metrics can be defined as follows:

$$precision = \frac{\sum_s h_s}{\sum_s n_s} \qquad recall = \frac{\sum_s h'_s}{\sum_s |P_{\tau_s}|} \tag{36}$$

where n_s is the number of displays in session s, and P_{τ_s} is the set of purchased items in session s.

In addition, we report the Normalized Discounted Cumulative Gain (NDCG), which is sensitive to the position at which a hit occurs (unlike precision and recall). This property is important because of the so-called *position bias* - the human's natural tendency to examine ranked lists from top to bottom, giving higher weight to the topmost items. From the perspective of user experience, algorithms optimized for top-weighted lists are advantageous because they maximize the chance of a user seeing the most relevant content from first glance at recommendations, thus reducing the time needed to find the item of interest, and minimizing the probability of missing it. To account for the position bias, NDCG first computes the discounted sum of relevance scores of the ranked items (DCG). The discount is nothing but a decreasing function of the item's rank. As Equation 37 shows, the discount in NDCG is logarithmic. The relevance score in our case reflects the binary reward for the displayed recommendation at a given rank j, i.e. $r_j \in \{0, 1\}$. The DCG is then normalized by the DCG of the ideal ranking (i.e. when all displayed items yield positive rewards), denoted as IDCG.

For a set of queries Q, NDCG at rank N is computed as follows:

$$NDCG = \frac{1}{|Q|} \sum_{q=1}^{|Q|} \frac{DCG_q}{IDCG_q}$$

where

$$DCG_q = \sum_{i=1}^{N} \frac{r_i}{\log_2(i+1)}, \quad IDCG_q = \sum_{i=1}^{\min(|P|,N)} \frac{1}{\log_2(i+1)} \tag{37}$$

All our experiments report the averages over 10 runs of the simulation.

8.8.3 Experiment 1: standard vs. modified Thompson Sampling

In our first experiment, we compare the modified Thompson Sampling policy adapted for non-stationary and "sleeping" actions (i.e. Algorithm 6) to the standard Thompson Sampling algorithm. The comparison table is given below. We note that TS(SB+DP) in Table 19 is equivalent to BEER[TS] from Algorithm 6.

Table 19: Standard vs. modified Thompson Sampling

Agent	Recall	Precision	NDCG
		Books	
TS (SB+DP)	**0.0581** ↑83.3%	**0.0210** ↑94.4%	**0.0716** ↑94.6%
TS (SB)	0.0562 ↑77.3%	0.0202 ↑87.0%	0.0660 ↑79.3%
TS	0.0317	0.0108	0.0368
		Fashion	
TS (SB+DP)	0.1734 ↑21.3%	**0.0152** ↑13.4%	**0.0483** ↑19.9%
TS (SB)	**0.1760** ↑23.2%	0.0142 ↑6.0%	0.0427 ↑6.0%
TS	0.1429	0.0134	0.0403
		Yoochoose	
TS (SB+DP)	0.2973 ↑10.5%	**0.0419** ↑10.8%	**0.1391** ↑21.4%
TS (SB)	**0.2984** ↑10.9%	0.0409 ↑8.2%	0.1273 ↑11.1%
TS	0.2690	0.0378	0.1146

SB = Sleeping Bandit, DP = Dynamic Partitioning

Table 19 reveals that accounting for "sleeping" actions in Thompson Sampling is a necessary measure for our application, since it improves the accuracy on all three metrics and all datasets. The "sleeping" mode becomes especially important when frequently chosen components cannot answer many queries, in which case they should allow other (responding) components to take over the query.

It can be seen that component partitioning is able to further boost the accuracy on all datasets in terms of precision and NDCG ranking. In the Books dataset, it also improves recall. In the other two datasets, we observe the typical precision-recall trade-off, with recall suffering small losses yet remaining within comparable value ranges. This can be explained by the fact that sub-components have limited catalog coverage (as compared to an unpartitioned component), and hence can miss some items in the purchase order P_τ. However, they increase the likelihood of a relevant item being purchased by improving its exposure (precision) and ranking (NDCG) in recommendation lists. We also note that in this experiment we applied the same thresholds for component partitioning in all datasets, and hence it would be reasonable to expect higher scores with dataset-specific thresholding strategies. We leave this analysis for the future.

We performed a two-tailed t-test on the results in Table 19 and it proved them to be statistically significant at an alpha level of 0.01.

8.8.4 Experiment 2: BEER(TS) vs. baselines

We now compare our ensemble recommendation algorithm against two stand-alone recommenders, namely *best sellers* (BS) and *those-who-bought-also-bought* (TWBAB). Both predictors are based on item (co-)purchases, which is clearly the strongest behavioral signal in a non-personalized context. This makes these predictors highly competitive as baselines for our experiment. While making this comparison, we evaluate two variants of BEER[TS] - *with* and *without* the aforementioned baselines as components of the ensemble. In this comparison, it is also worth looking at the coverage of queries by the recommenders. Table 20 summarizes the results of the experiment.

Regardless of whether the two baselines are part of the ensemble or not, BEER[TS] vastly outperforms the stand-alone baselines in terms of recall and NDCG. It is also clear from Table 20 that TWBAB tends to be the most *precise* recommender (for Books and Yoochoose). However, this predictor has rather poor query coverage, which is especially evident in the Books dataset, with only 41.64% query response rate. The query coverage of BEER[TS] without baselines is also lower than in other datasets, which is why the Books dataset benefits more from the "sleeping" bandit configuration in the previous experiment (Table 19).

The *best sellers* baseline, on the other hand, is capable of answering

Table 20: Ensemble recommender vs. stand-alone baselines

Agent	Recall	Precision	NDCG	query coverage
		Books		
BEER[TS] incl. baselines	**0.0589**	0.0213	**0.0719**	**1.0**
BEER[TS] excl. baselines	0.0551	0.0275	0.0655	0.7185
TWBAB	0.0324	**0.0365**	0.0366	0.4164
Best sellers	0.0105	0.0085	0.0292	**1.0**
		Fashion		
BEER[TS] incl. baselines	**0.1750**	**0.0155**	**0.0504**	**1.0**
BEER[TS] excl. baselines	0.1704	0.0149	0.0474	0.9825
TWBAB	0.1185	0.0136	0.0277	0.7792
Best sellers	0.0490	0.0119	0.0374	**1.0**
		Yoochoose		
BEER[TS] incl. baselines	**0.2985**	0.0424	**0.1424**	**1.0**
BEER[TS] excl. baselines	0.2953	0.0470	0.1376	0.8999
TWBAB	0.2311	**0.0519**	0.1063	0.7286
Best sellers	0.1211	0.0234	0.0687	**1.0**

TWBAB = those-who-bought-also-bought

every query (past the very first session), since its recommendations are not query-specific. Therefore, the inclusion of this baseline makes it a good fallback method for the ensemble learner. With the addition of TWBAB, the learner can benefit from TWBAB's high precision in cases when it gives an answer. Indeed, we observe that the inclusion of the two baselines to BEER[TS] results in better recommendation accuracy with an excellent query coverage.

8.8.5 Experiment 3: MAB policies within BEER

As noted earlier, it is easy to replace Thompson Sampling with other MAB policies within the BEER framework. In this experiment, we run several instances of our ensemble learner powered by various MAB policies. Apart from TS, we consider five variants of UCB algorithms, namely UCB1 [14], UCB-Tuned [14], UCB-Bayes [135], KL-UCB [87], and MOSS [13], as well as the pre-tuned ϵ-greedy policy. The simplest of these policies is ϵ-greedy, which chooses an arm greedily with probability $1-\epsilon$, and plays a random arm otherwise. A well-tuned ϵ-greedy is the obvious first choice as a baseline bandit policy. The Upper Confidence Bound (UCB) family of algorithms is clearly one of the most popular solutions to MAB problems. The key difference of this policy from ϵ-greedy and Thompson Sampling is that it is deterministic (in its original formulation). It chooses an arm based on a confidence interval, within which the expected reward is to be found. The policy makes optimistic guesses based on the upper bound of this interval (UCB). The arm which maximizes the sum of expected reward and upper confidence bound is pulled. If the guess is wrong, then the upper bound will quickly decrease and we will start exploring other arms. If the guess is right, the chosen action will be exploited, thus contributing to the reduction of regret. This is how UCB policies balance exploration and exploitation. Over time, repeated arm pulls will eventually approach the true expected value. Different variants of this strategy that are used in our experiment differ in how they estimate the UCB and/or put constraints on the reward distribution.

The resulting scores for different policies are summarized in Table 21.

We observe that Thompson Sampling maintains its top rank for all metrics in all three datasets. In line with the findings of Gopalan et al. [91], we conclude that Thompson Sampling is very effective in solving complex practical online tasks, such as e-commerce recommendations. Our experimental findings support the claims that were made in Section 8.7.2 regarding the advantages of Thompson Sampling over UCB-style policies in situations such as delayed reward attribution, which is our case here. The strength of Bayesian policies is also evident from the fact that UCB-Bayes is the second best performer.

8.8.6 Experiment 4: priming the sampler

Having established the best policy for orchestrating the ensemble components, we experiment with priming the Thompson Sampler with

Table 21: Ensemble learner with different MAB policies

Agent	Recall	Precision	NDCG
	Books		
BEER[TS]	**0.0589**	**0.0213**	**0.0719**
BEER[UCB-Bayes]	0.0578	0.0210	0.0709
BEER[UCB-Tuned]	0.0568	0.0206	0.0694
BEER[MOSS]	0.0567	0.0205	0.0694
BEER[UCB1]	0.0526	0.0188	0.0642
BEER[KL-UCB]	0.0500	0.0195	0.0657
BEER[ϵ-greedy]	0.0024	0.0117	0.0382
	Fashion		
BEER[TS]	**0.1750**	**0.0155**	**0.0504**
BEER[UCB-Bayes]	0.1701	0.0147	0.0480
BEER[ϵ-greedy]	0.1697	0.0151	0.0489
BEER[MOSS]	0.1656	0.0141	0.0463
BEER[KL-UCB]	0.1630	0.0149	0.0485
BEER[UCB-Tuned]	0.1602	0.0133	0.0437
BEER[UCB1]	0.1426	0.0114	0.0377
	Yoochoose		
BEER[TS]	**0.2984**	**0.0424**	**0.1424**
BEER[UCB-Bayes]	0.2976	0.0422	0.1414
BEER[UCB-Tuned]	0.2975	0.0421	0.1412
BEER[MOSS]	0.2971	0.0422	0.1413
BEER[UCB1]	0.2955	0.0415	0.1394
BEER[KL-UCB]	0.1794	0.0287	0.0950
BEER[ϵ-greedy]	0.1211	0.0234	0.0755

side information in attempt to give it an additional accuracy boost. As outlined in Section *Sampler Priming*, this information may come from two sources: a) pre-existing user sessions; and b) the product catalog.

Priming with pre-recorded event data The deployment of the recommender system on an existing website with some history of past user sessions makes it possible to prime the sampler in two ways. First, the behavioral components can be warm-started with initial item associations from past events. This allows the sampler to take advantage of high-precision behavioral components from the very start. Second, after running the sampler through historical data we obtain the initial posterior distributions for each component, which allows us to assign informative priors to the Thompson Sampler. We evaluate both scenarios by running BEER[TS] through pre-recorded 100K events preceding our test data for each dataset. In Table 22 we evaluate the effect of pre-training behavioral components and priming the component's Beta distributions separately.

Table 22: Priming the sampler with pre-recorded event data

Agent	Recall	Precision	NDCG
	Books		
BEER[TS]: pre-trained+primed	0.0632 ↑7.3%	**0.0231** ↑8.5%	**0.0828** ↑15.2%
BEER[TS]: pre-trained	**0.0633** ↑7.5%	0.0230 ↑8.0%	0.0822 ↑14.3%
BEER[TS]: cold-start	0.0589	0.0213	0.0719
	Fashion		
BEER[TS]: pre-trained+primed	**0.1805** ↑3.2%	**0.0159** ↑2.6%	**0.0517** ↑2.8%
BEER[TS]: pre-trained	0.1792 ↑2.5%	0.0157 ↑1.3%	0.0511 ↑1.6%
BEER[TS]: cold-start	0.1749	0.0155	0.0503
	Yoochoose		
BEER[TS]: pre-trained+primed	**0.3115** ↑4.5%	**0.0434** ↑3.1%	0.1464 ↑3.2%
BEER[TS]: pre-trained	0.3096 ↑3.9%	**0.0434** ↑3.1%	**0.1466** ↑3.4%
BEER[TS]: cold-start	0.2980	0.0421	0.1418

It becomes clear that pre-training behavioral components has a noticeable effect on the accuracy of recommendations (even when the pre-recorded dataset is 5 times smaller than the test set). However, the effect of priming the Beta parameters is less evident. This especially holds for datasets with few components, such as Yoochoose.

Catalog-based priming A more challenging priming scenario is when the only available side information is the product catalog. Even though priming behavioral components in this situation is not feasible, we may still make informed guesses about the predictive ability of attribute-based components. One way of reasoning about their means is to make them inversely proportional to the average query response size. This has the same intuition we used for IDF-based partitioning (Section 8.6): attributes that yield larger sets of recommendable items for a given query (i.e. the currently examined item) have lower expected rewards as compared to attributes that produce 'sharper' responses. In other words, assuming that each query response from action a contains one relevant item, the mean μ_a is equal to the probability of that item appearing in the top-1 result. We can express it as follows:

$$\mu_a = \frac{1}{|V(G_a)|} \sum_{q \in V(G_a)} \frac{1}{|N_{G_a}(q)|} \tag{38}$$

where $V(G_a)$ is a set of vertices (items) connected via attribute a, and $N_{G_a}(q)$ is a neighborhood of query item q containing items with the same value of attribute a as item q.

The calculation of the variance is done via Equation (34), where we need to set the uncertainty parameter ν according to our confidence in the estimation of μ_a. We envisage several possibilities:

1. Uniform ν. For the standard Thompson Sampling with priors $\alpha = 1, \beta = 1$, we have: $\nu = \alpha + \beta + 1 = 3$. This corresponds to the default variance of $1/12$.

2. Component-specific ν depending on the component's query coverage. For example, $\nu_a = \sqrt{|V(G_a)|} + 1$. Intuitively, the more items (i.e. queries) have contributed to estimating the means, the more reliable that estimate is. Plugging this ν_a to Equation (34) will result in variances reflecting our degree of uncertainty in the primed means.

Another factor worth considering is posterior scaling. In classical Thompson Sampling, all actions start off with uniform priors with $\mu = 0.5$. In most recommendation scenarios, however, it is reasonable to assume that the true Beta means will be much closer to 0. This can be seen in Table 20, where the precision of the best stand-alone predictor (TWBAB) is at best 0.0519. Therefore, we may aid convergence by setting the initial means to lower values, such as 0.05. In case the distributions are not uniform but primed, we can scale the primed means to the interval [0,0.05].

Table 23: Catalog-based priming

Agent	Recall	Precision	NDCG
Books			
BEER[TS]: catalog-primed, $\nu_a = \sqrt{\lvert V(G_a)\rvert} + 1$	**0.0592**	**0.0214**	**0.0721**
BEER[TS]: catalog-primed, $\nu_a = 3$	0.0587	0.0213	0.0718
BEER[TS]: cold-start	0.0587	0.0212	0.0716
Fashion			
BEER[TS]: catalog-primed, $\nu_a = \sqrt{\lvert V(G_a)\rvert} + 1$	**0.1761**	0.0156	0.0506
BEER[TS]: catalog-primed, $\nu_a = 3$	0.1752	**0.0157**	**0.0508**
BEER[TS]: cold-start	0.1750	0.0155	0.0503

We now evaluate the catalog-based priming of attribute-based components based on Equation (38) and the two aforementioned variance estimation strategies. Behavioral components cannot be primed in this case and hence maintain their uniform distributions, but are scaled as explained above. This time we exclude the Yoochoose dataset from the analysis, since it only has 2 attribute-based components. The results for the two other proprietary datasets (having 45 and 51 attribute-based components) are presented in Table 23.

Our results indicate that priming the sampler in the absence of any historical event data is indeed feasible, but the additional accuracy gain obtained in our experiments is negligible (below 1%). While other approaches for the estimation of primed means might yield slightly better results, we conclude that adjusting prior parameters is not crucial for the convergence of Thompson Sampling, as the algorithm learns very fast anyway. However, what helps to warm-start the algorithm, as is shown in Table 22, is pre-training behavioral components using past event data.

8.8.7 Experiment 5: Session-based personalization

In our last experiment, we add the kNN component to the ensemble, thus making it able to deliver personalized recommendations within the scope of a session. In its stand-alone version, it serves as a strong baseline that would be interesting to run next to our ensemble in complement to experiment 2. We compare these two configurations (e.g. kNN stand-alone and kNN within BEER[TS]) with our standard configuration that uses only elementary components.

Parameter tuning The kNN parameters discussed in Section 8.4.3 were tuned on a disjoint validation set of 500K events preceding the events of the test set. These parameters with their respective value ranges are presented in Table 24. Table 25 reports the optimal values of the parameters for the 3 datasets. These values were used to parameterize the kNN component in order to optimize its performance in our experiments.

Table 24: kNN parameters

Parameter	Description	Range of values
k	number of neighbors	10 - 1000
$w_{sim} : (w_{click}, w_{cart}, w_{purchase})$	similarity weights	0.0 - 1.0
$w_{rec} : (w_{click}, w_{cart}, w_{purchase})$	recommendation weights	0.0 - 1.0
hlife	event half-life	0 - 100
session_hlife	session half-life	0 - #*sessions*
similarity	similarity measure	cosine euclidean jaccard dice log-likelihood

Notably, equal weighting of event types for the similarity computation yielded the best results for all datasets. We also observe that for Books dataset, having no ageing at all is preferable.

Table 25: Optimized kNN parameters

Parameter	Books	Fashion	Yoochoose
k	100	100	100
$w_{sim} : (w_{click}, w_{cart}, w_{purchase})$	(1.0,1.0,1.0)	(1.0,1.0,1.0)	(1.0,1.0,1.0)
$w_{rec} : (w_{click}, w_{cart}, w_{purchase})$	(0.5,0.7,1.0)	(0.5,0.7,1.0)	(1.0,1.0,1.0)
hlife	∞	∞	25
session_hlife	∞	#sessions	10000
similarity	jaccard	cosine	dice

Pre-preprocessing In order to demonstrate the impact of pre-processing in kNN (as described in Section 8.4.3), we report running times and accuracy scores of two kNN runs: *with* and *without* the pre-processing step. The results are given in Table 26.

Table 26: The effect of pre-processing in kNN

Agent	Recall	Precision	NDCG	Runtime
		Books		
kNN with pre-processing	**0.0626** ↑14.7%	**0.0391** ↑51.0%	0.0871 ↓1.6%	**00:03:15**
kNN w/o pre-processing	0.0546	0.0259	**0.0885**	01:09:58
		Fashion		
kNN with pre-processing	**0.1726** ↑61.3%	0.0171 ↓1.2%	0.0558 ↓0.2%	**00:06:58**
kNN w/o pre-processing	0.1070	**0.0173**	**0.0559**	01:19:27
		Yoochoose		
kNN with pre-processing	**0.3120** ↑13.4%	**0.0485** ↑5.9%	**0.1606** ↑2.2%	**00:07:13**
kNN w/o pre-processing	0.2751	0.0458	0.1571	02:09:38

The above results make it obvious that the pre-processing step is crucial in kNN, as it drastically improves not only the running time of the algorithm, but its accuracy as well. The observed boost in recall using the proposed pre-processing procedure is indeed very evident. This especially holds for the Fashion dataset, that is characterized by long sessions (twice as long as the other two datasets). In such cases, the overlapping sessions are more "noisy" and may not capture the intent of the user well. Explicitly biasing the algorithm in favor of sessions containing the current query item appears to be a good strategy for filtering out the multitude of noisy neighbors.

Personalizing the ensemble We now demonstrate how the ensemble is affected by the inclusion of kNN, and how it compares with the stand-alone kNN recommender (Table 27).

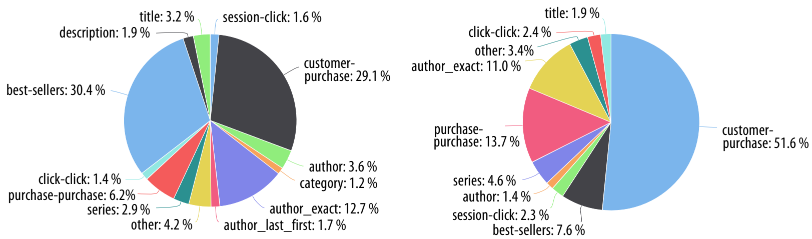
Table 27: BEER(TS) with kNN

Agent	Recall	Precision	NDCG	Coverage
		Books		
BEER[TS] with kNN	**0.0716** ↑31.1%	0.0279 ↑30.4%	**0.0966** ↑64.0%	1.0
kNN	0.0626 ↑14.7%	**0.0391** ↑82.7%	0.0872 ↑48.0%	0.6892
BEER[TS]	0.0546	0.0214	0.0589	1.0
		Fashion		
BEER[TS] with kNN	**0.1773** ↑1.0%	0.0168 ↑8.4%	0.0553 ↑9.5%	1.0
kNN	0.1725 ↓1.8%	**0.0171** ↑10.3%	**0.0558** ↑10.5%	0.9931
BEER[TS]	0.1756	0.0155	0.0505	1.0
		Yoochoose		
BEER[TS] with kNN	0.3099 ↑3.9%	0.0445 ↑5.0%	0.1521 ↑6.9%	1.0
kNN	**0.3120** ↑4.6%	**0.0485** ↑14.4%	**0.1606** ↑12.9%	0.9658
BEER[TS]	0.2983	0.0424	0.1423	1.0

It can be seen that kNN alone is capable to surpass the performance of the ensemble, especially with respect to precision and NDCG. This is manifested most prominently in Books dataset. This result is not surprising however, since the comparison is between a personalized recommender and a non-personalized one, where the odds are obviously in favor of the former. One can also notice that the query coverage of kNN is inferior to that of the ensemble. This is because the stand-alone kNN will fail to generate recommendations during the initial period of cold-start, when no session overlaps can be found.

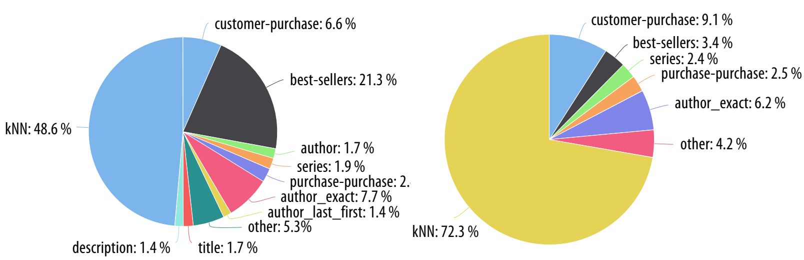
The inclusion of kNN to the ensemble leads to observable improvements on all metrics in all three datasets. In Books, the measured boost of 65% in NDCG and ≈30% in precision and recall is noteworthy, considering that this is an effect of adding a single component. In Fashion, we can notice that although the recall of the stand-alone kNN recommender is slightly worse than that of the ensemble, its integration to BEER[TS] improves recall over the basic configuration. This has to do with the fact that the addition of an extra component affects the interplay between components of the ensemble. This influence on the choice of components by the sampler has a positive effect when a high-precision component such as kNN is introduced. To better understand what is happening under the hood of the ensemble, we report component statistics in the next sub-section.

Component statistics We now present pie charts that demonstrate the exposure and success rate of the most significant components for each dataset (Figures 14-19).



(a) Exposure in % of 587660 displays **(b)** Successes in % of 13385 hits

Figure 14: Books dataset: component statistics for BEER(TS)



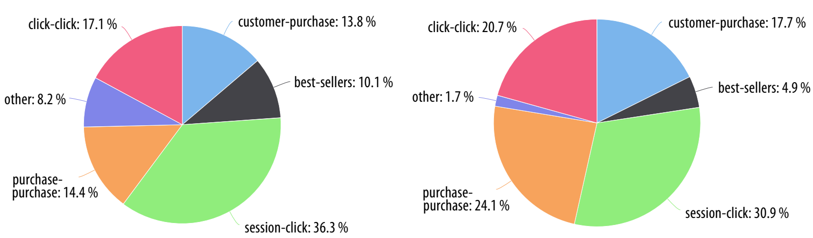
(a) Exposure in % of 587660 displays **(b)** Successes in % of 17505 hits

Figure 15: Books dataset: component statistics for BEER(TS) with kNN

Figure 14a shows the breakout of component displays in the standard configuration of BEER[TS] for the Books dataset. We observe that *best-sellers* and *customer-purchase* enjoy the highest exposure, having roughly equal share of displays (≈30%). We also note the appearance of several attribute-based components, such as *author*, *description*, *series*, and *title*, which manage to compete with stronger behavioral predictors. The *author_exact* component, in particular, does it quite well, with 11% success rate (Figure 14b). It appears that *customer-purchase* is responsible for the majority of hits in this dataset (51.6% success share).

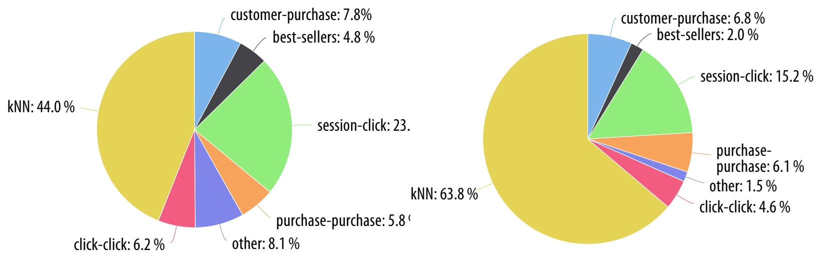
Figure 15a shows how the addition of kNN to the ensemble affects the exposure of other components. The impact of this inclusion is large, especially on *customer-purchase*, whose usage drops from 29.1% to 6.6%. Apparently, the reason for this large decline lies in the fact that these two components happen to have a large overlap in queries that they answer. Since the precision of kNN appears to be considerably higher (see Figure 15b), it ends up getting picked much more often by the sampler than *customer-purchase*.

The attribute-based components are affected in a lesser way, since they seem to specialize in different types of queries. This also holds for *best-sellers*, whose exposure remains fairly high. The reason is that it answers many 'cold-start' queries, with virtually no competition from other predictors. Accuracy-wise, we observe kNN's immense contribution of 72.3% to the overall hit rate of the ensemble (Figure 15b).



(a) Exposure in % of 870365 displays **(b)** Successes in % of 15272 hits

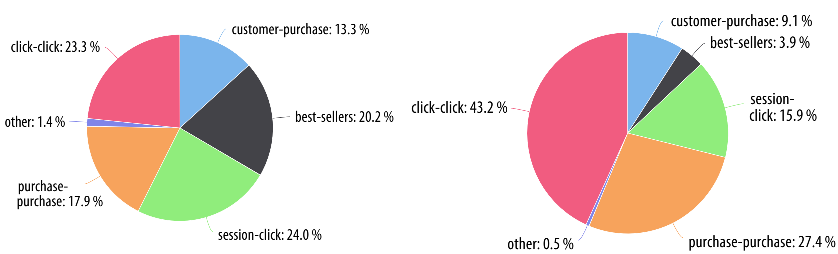
Figure 16: Fashion dataset: component statistics for BEER(TS)



(a) Exposure in % of 870365 displays

(b) Successes in % of 16457 hits

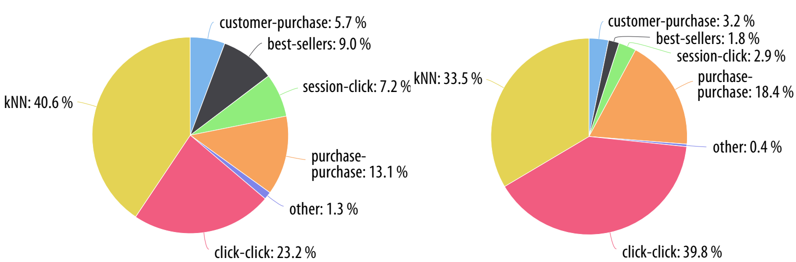
Figure 17: Fashion dataset: component statistics for BEER(TS) with kNN



(a) Exposure in % of 1074990 displays

(b) Successes in % of 47221 hits

Figure 18: Yoochoose dataset: component statistics for BEER(TS)



(a) Exposure in % of 1074990 displays

(b) Successes in % of 49653 hits

Figure 19: Yoochoose dataset: component statistics for BEER(TS) with kNN

Figure 16a depicts the exposure of components in the standard ensemble for the Fashion dataset. We immediately notice the absence of any strong attribute-based predictors, despite that their amount in this dataset is the largest of the three datasets analyzed. Instead, we observe their cumulative exposure of 8.2%, which is less than of any one behavioral component. Furthermore, their hit contribution in this dataset is negligible at 1.7% (Figure 16b).

The inclusion of kNN follows the similar pattern previously observed on the Books dataset, with substantial suppression of behavioral predictors, but no impact on attribute-based ones (Figure 17a). The dominance in success rate is also very evident at 63.8% (Figure 17b).

In Figure 18a, we observe the nearly uniform distribution of (behavioral) component displays in the standard configuration of the ensemble on the Yoochoose dataset (we remind the reader that there are only 2 attributes present in the dataset). The distribution of successes, however, is more skewed, with the largest contribution coming from the *click-click* component (43.2%, see Figure 18b).

Contrary to what we see on other datasets, the exposure of this dominant behavioral component is not affected by the inclusion of kNN (Figure 19a). In fact, *click-click* manages to maintain its dominance in success rate (Figure 19b), and the two components appear to operate in good synergy by answering different types of queries. For instance, *click-click* will fail on queries where the query item has no history of 'co-clicks'. Such a case, however, is not problematic for the kNN component, who will still be able to find meaningful neighboring sessions with high probability.

This dataset also shows the characteristic behavior of the *best-sellers* component (Figure 18), whose low hit rate (3.9%) does not seem to justify its high exposure (20.2%). This is the price to pay for the perfect query coverage of the ensemble, who chooses *best-sellers* when no other predictor can handle the query.

8.9 Conclusions

8.9.1 Contribution summary

E-commerce is one of the most important and challenging application domains of recommender systems, where recommendations must be adaptively computed for each new product page visit. To address this problem, we proposed an item-to-item ensemble recommendation algorithm with Thompson Sampling at its core. A user interface that serves intelligent recommendations to customers of an e-commerce store in real-time makes it more likely that they will encounter products of interest as they explore the product catalog. Furthermore, the proposed framework allows system developers to deliver *semantically transparent user interfaces* [148], where each recommendation is accompanied with a clear explanation of its source (e.g. behavioral or attribute-based; see Figure 11). The explainability of recommendations has proved to be of paramount importance to users, as it encourages them to interact with the system and correct wrong impressions [153], instead of having to face a 'black box'. The proposed algorithm is also friendly to new users of the system, who can benefit from recommendations literally from the first product page visit, since the ensemble framework does not rely on user profiles or individual browsing patterns. However, if such a pattern is already in place, the existence of a session-based personalized component such as kNN can boost the accuracy of recommendations further. We provide our approach to kNN featuring event weighting and time decay for e-commerce session data.

The proposed algorithm was designed with a commercial application in focus, making it simple for a potential vendor to introduce recommendation functionalities to new or existing e-commerce platforms. The recommendation components of the ensemble can be constructed automatically from the existing item attributes or user-item interaction types (collaborative signals). At the same time, the system allows vendors to effortlessly integrate hand-crafted components into the ensemble, which are immediately picked up and explored by the Thompson Sampler. Such a component-based architecture is scalable, since the number of bandit arms does not depend on the number of items. An integral part of our recommendation bandit is its adaptation to realistic situations with "sleeping" actions and non-stationary reward sequences through dynamic partitioning of recommendation components.

In a series of experiments on real e-commerce session data, we show that the above adaptations are crucial for the algorithm's operation (with accuracy gains ranging from 83% to 94% for the Books dataset). We further evaluate the strength of the ensemble approach, achieving far superior results in comparison to strong stand-alone baseline recommenders, namely *best-sellers* and *those-who-bought-also-bought*. As we have observed, adding these baselines to the ensemble does not only improve its accuracy, but also attains 100% query coverage. We also show how this non-personalized ensemble compares to a personalized recommender (kNN), and demonstrate the accuracy gain from introducing session-based personalization. Our comparative study convinces us that Thompson Sampling is empirically the best MAB policy for the BEER framework. We further show that Thompson Sampling handles cold-start gracefully out-of-the-box, as it does not seem to benefit much from Beta parameter priming. This property together with TS's robustness to observation delays makes it a very attractive method for attacking the recommendation problem using reinforcement learning. Our final experiment illustrates how recommendation components compete for exposure. This has a direct effect on the appearance of the recommendation interface to the user: will it become more of a navigational tool for exploring similar items based on shared attributes, or will it behave more like a discovery mechanism for popular or serendipitous goods, when collaborative filters are at work? Since in Thompson Sampling, exposure is a function of success, user behavior itself will play its role in shaping the character of an adaptive recommender system.

8.9.2 Future work

The proposed ensemble algorithm is as good as the components that it is comprised of. It is therefore worth exploring better strategies for constructing base predictors for the ensemble. When it comes to behavioral signals, we expect that the user-based (as opposed to session-based) personalization would bring additional benefits to the ensemble, provided that user profile data is available. This way we can take advantage of both short-term and long-term user preferences when computing recommendations. With regard to attribute-based components, the next research direction is to develop methods for the automatic composition of high-precision attribute combinations as potential components of the ensemble. In addition, it is worth considering components built from item descriptions and/or images, thus utilizing the full range of data available in an e-commerce application. At the same time, effort should be made to retain the explainability of the resulting ensemble method at an acceptable level, making sure that the user-system interaction is as transparent as possible.

9 PAPER IV

Paraschakis D. and Nilsson B.J.

"`FlowRec`: Prototyping Session-based Recommender Systems in Streaming Mode"

Abstract Despite the increasing interest towards session-based and streaming recommender systems, there is still a lack of publicly available evaluation frameworks supporting both these paradigms. To address the gap, we propose `FlowRec` — an extension of the streaming framework `Scikit-Multiflow`, which opens plentiful possibilities for prototyping recommender systems operating on sessionized data streams, thanks to the underlying collection of incremental learners and support for real-time performance tracking. We describe the extended functionalities of the adapted prequential evaluation protocol, and develop a competitive recommendation algorithm on top of `Scikit-Multiflow`'s implementation of a Hoeffding Tree. We compare our algorithm to other known baselines for the next-item prediction task across three different domains.

Published in Proceedings of the 24th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), 2020

9.1 Introduction

In the past few years, the RecSys community has witnessed a paradigm shift from the traditional matrix completion problem to sequential *session-based* recommendations [126, 232]. The latter approach is dominated by neural methods that are often evaluated in an online manner, i.e. when the events of a session are sequentially revealed and predicted one-by-one. However, these systems are still trained in batches on large chunks of recorded data [108, 232, 280]. To better approximate real-world scenarios, *streaming* recommender systems [63, 275, 296] have been designed for incremental online learning from continuous data streams in the context of limited memory/runtime, and anytime prediction [272]. However, most of them address the conventional rather than session-based recommendation problem [96]. Bridging the gap between session-based and streaming recommender systems has been recently attempted [96, 126], marking an emerging research direction of a high practical value.

Presently, only a few publicly available benchmarking frameworks for streaming recommendations exist. Some of them have been designed for a specific application domain [126, 264], while others lack native support for session data [83, 140]. `Scikit-Multiflow` [198] has recently been released as a general-purpose Python framework for stream mining, offering a variety of stream learners, change detectors, and evaluation methods. To facilitate the research on streaming session-based recommendations, we propose `FlowRec`[29] - an extension of `Scikit-Multiflow` for rapid prototyping of recommender systems. Currently, it contains several stream-oriented recommenders and metrics for prequential evaluation. Additionally, we demonstrate a principled way of exposing a recommendation interface to an underlying stream learner class of `Scikit-Multiflow` (namely, a Hoeffding Tree). We show that the resulting recommender system has remarkable performance against established baselines. `FlowRec`'s functionality is detailed in the next section.

[29] **https://git.io/flowrec**

9.2 FlowRec

The framework consists of three main entities: a stream, an evaluator, and a model. This section describes the interplay between these entities.

9.2.1 Problem setting

Consider a stream D of (overlapping) user sessions $S = s_1, \ldots, s_{|S|}$ (Figure 20). A session represents an ordered sequence of events of the form (X, y), where X is a feature vector describing the context for item y. As a bare minimum, X contains the session identifier for the item. Other common features are timestamp and event type (e.g. click, purchase, etc.). The scope of our study is limited to the context of collaborative filtering, which relaxes the assumption of item metadata in feature vectors (technically, any feature can be encoded as a part of X).

At each time step $t = 1, \ldots, T$, the stream provides a sample (X, y). Based on the information in X, the model is asked to generate a list of N predictions $\hat{Y} = (\hat{y}_1, \ldots, \hat{y}_N)$ in an attempt to correctly guess the hidden item y. This corresponds to the *next-item prediction task* in the RecSys literature [232]. In practice, only certain features of X are retained for the prediction part, such as the current session identifier, and possibly the timestamp of the event. After the prediction, the entire feature vector X together with the label y are revealed to the model, allowing it to make an incremental update. This iterative, supervised 'test-then-train' methodology is known as *prequential evaluation* [296] (Figure 21).

In our framework, each processed sample (X, y) is added to the sliding window of the last n observations, which we refer to as *observation window* (depicted as the green box in Figure 20). Although its use is not required in the ordinary stream learning, it can ease the development of session-based models by providing a snapshot of the recent session data on demand. The size of the window must be chosen in consideration of the system's memory and runtime constraints.

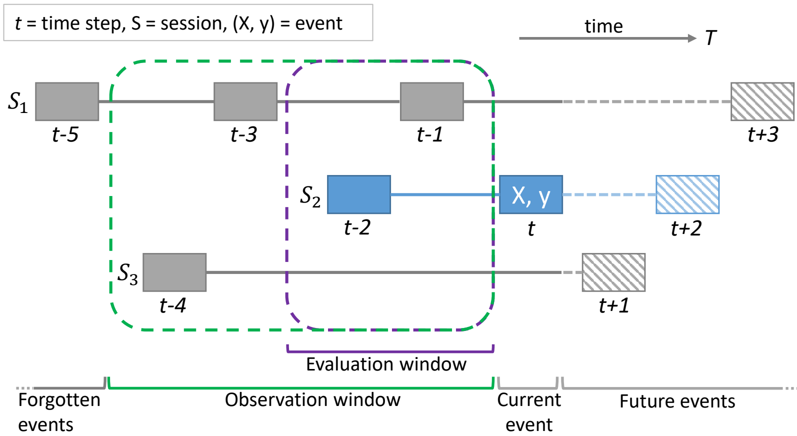


Figure 20: Streaming sessions

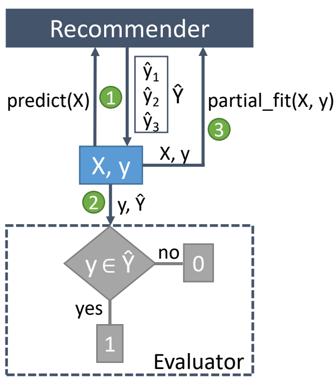


Figure 21: Prequential protocol

9.2.2 Metrics

FlowRec implements several evaluation metrics, two of which are commonly used for next-item prediction [96, 108, 173, 280], namely *recall* (a.k.a. hitrate) and *mean reciprocal rank* (MRR). Recall measures the average number of successful predictions, whereas MRR measures their average reciprocal ranking, i.e.:

$$Recall@N = \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{N}\mathbb{1}(y_t = \hat{y}_i) \tag{39}$$

$$MRR@N = \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{N}\frac{1}{i}\mathbb{1}(y_t = \hat{y}_i) \quad (40)$$

The framework keeps two sets of measurements: (a) *global*, where the running average of each metric is calculated from all the past data; and (b) *sliding*, where the average is taken over a sliding window of recent events that we call the *evaluation window* (purple box in Figure 20). The sizes of evaluation and observation windows are user-adjustable, offering flexibility in simulation setups.

9.2.3 Prequential evaluation

The basic workflow for measuring recall using prequential evaluation is presented in Algorithm 7. The complete functionality of `Scikit-Multiflow`'s prequential evaluator is provided in its official documentation[30].

`FlowRec` introduces the following additional parameters[31] for the evaluator:

- Indices of data columns holding session, timestamp, and event type identifiers. The last two columns are optional, and allow for time-aware and event-specific training and/or evaluation.
- Stream-related configurations, such as the size of the observation window, and the number of events to skip from the start of the stream.
- Recommendation-specific settings, such as the size of the recommendation list, and the event types that trigger recommendation requests. There are also flags for enabling/disabling *reminders* and *repeated* recommendations. Reminders are recommendations of items that were *visited* earlier by the user, whereas repeated recommendations are those that were already *given* earlier to the user.

[30] **https://scikit-multiflow.github.io/scikit-multiflow/documentation.html**

[31] **https://flowrec.readthedocs.io/en/latest/eval_parameters.html**

The size of the evaluation window (as well as other settings) are covered by the original parameter list of the `EvaluatePrequential` class in `Scikit-Multiflow`. Note that the stream provides samples in the order as they appear in the dataset. The last column of the dataset should always contain the item identifiers.

Algorithm 7 Basic prequential protocol for measuring recall

Input: D: data stream, N: recommendation cutoff, n_{keep}: size of the observation window, M: set of recommendation models

Output: Recall@N of each model

$W \leftarrow create\_queue(n_{keep})$ ▷ observation window of n_{keep} latest events
$r_m \leftarrow 0, \forall m \in M$ ▷ reward counter
$n \leftarrow 0$ ▷ evaluation counter
while $D.has\_more\_samples()$ **do**
  $X, y \leftarrow D.next\_sample()$
  if $X.session \in W.sessions$ **then**
    for each $m \in M$ **do**
      $\hat{Y} \leftarrow m.predict(X)$
      if $y \in top\_N(\hat{Y})$ **then**
        $r_m \leftarrow r_m + 1$
    $n \leftarrow n + 1$
  for each $m \in M$ **do**
    $m.partial\_fit(X, y)$
  $W.add(X, y)$
return $r_m/n, \forall m \in M$

9.3 Prototyping

Prototyping incremental session-based recommendation algorithms in `FlowRec` is straightforward. First, a streaming model is built by extending the `BaseSKMObject` class of `Scikit-Multiflow` with the appropriate mixin. It is natural to treat the recommendation task as a multi-class classification problem, where each class corresponds to an item. Hence, the suitable mixin for this type of problems is `Classifi-`

`erMixin`[32]. What remains is to implement the following abstract methods:

`partial_fit(X, y)` — Incrementally train a stream model.

`predict(X)` — Generate top-N predictions for the target's class.

`predict_proba(X)` — Calculate the probabilities of a sample pertaining to each of the available classes (implementing this method is optional).

Every stream model developed in `FlowRec` has access to useful shared resources, such as observation window, current session vector, and item catalog.

9.3.1 Session-based streaming models

Presently, `FlowRec` contains the following streaming models for session-based recommendations:

Rule-based models The first three models are rule-based recommenders that capture one-to-one relationships between items. These methods rely on the very last item of a session to make their predictions for the next item. Despite their simplicity, rule-based models have proven surprisingly effective in the domains of music and e-commerce [173], and have very low computational complexity. We briefly outline these methods below (refer to [173] for details).

Association Rules (AR) The rules are derived from co-occurrences of two items in a session (e.g. 'those-who-bought-also-bought'). A co-occurrence forms rules in both directions, i.e. $y_t \leftrightarrow y_{t'}, t \neq t'$.

Markov Chains (MC) The rules are derived from a first-order Markov Chain, describing the transition probability between two *contiguous* events in a session, i.e. $y_t \rightarrow y_{t+1}$.

[32] **`https://scikit-multiflow.github.io/scikit-multiflow/user-guide.core-concepts.architecture.html`**

Sequential Rules (SR) The rules are derived from sequential patterns between two items in a session, but not necessarily in successive events, i.e. $y_t \rightarrow y_{t'}, t' > t$. The sequential association is assigned the weight $1/(t' - t)$.

In `FlowRec`, the above algorithms can be made event-specific by providing the `event_column_index` parameter to the evaluator. For instance, predictors of type $purchase \leftrightarrow purchase$, $click \rightarrow purchase$, etc. can be useful in e-commerce applications as parts of an ensemble recommender [32].

Session kNN This is a specialized version of the k-Nearest Neighbors (kNN) algorithm that operates on session data. It is a strong baseline with performance comparable to that of certain deep neural methods [120, 173]. Being model-free, the algorithm is incremental by nature. S-kNN recommends items from other user sessions that are similar to the current session (a.k.a. neighbors). Given the active session S, the score of the candidate item $\hat{y}$ is calculated as follows:

$$score(\hat{y} \mid \hat{y} \notin S) = \sum_{S' \in \mathcal{N}(S)} sim(S, S') \cdot \mathbb{1}(\hat{y} \in S') \tag{41}$$

where $\mathcal{N}(S)$ is the k-sized neighborhood of session S, and $sim(S, S')$ is a measure of similarity between two sessions.

`FlowRec` implements Cosine, Jaccard, Dice, and Tanimoto similarity. In the future, we plan to implement other kNN variants described in [173].

BEER(TS) This is a bandit ensemble designed for streaming recommendations, which employs Thompson Sampling for the model selection. In [32], event-specific rule-based models were used as behavioral components of the ensemble. In `FlowRec`, any model implementing the `predict_proba` method can be added as a component of BEER[TS]. For each recommendation slot, the ensemble picks a model m with the highest sample $\theta_m \sim Beta(\alpha_m, \beta_m)$, where α_m is the number of past successes, and β_m is the number of past failures. The method is adaptable to non-stationary data (e.g. occurring due to concept drift), which is achieved via exploration-exploitation. In addition, BEER[TS] supports component splitting into (relatively) stationary partitions (see [32] for details), which is achieved in `FlowRec` by setting the boundaries for probabilities returned by the `predict_proba` method. Further, the components of the ensemble can complement each other in case of poor coverage, which helps to attack sparsity and cold-start issues.

Popularity baseline This is the traditional baseline that outputs the top-N most popular items in descending order.

The above methods can be incrementally trained either on the global scale, or within the observation window. The latter option acts as a forgetting mechanism for older data, which aids model scalability and adaptability to recent trends.

9.3.2 Hoeffding Tree wrapper

Prototyping streaming session-based recommenders can be facilitated by employing the rich collection of incremental algorithms offered by `Scikit-Multiflow`, including Bayesian methods, lazy learners, ensembles, neural networks, tree-based methods, and more [198]. Utilizing any of these methods for recommendation tasks is achieved via a *wrapper*, which is a middle layer that handles the inputs and the outputs of an underlying learner. Some of the common tasks performed by wrappers include:

- transforming a sample to the desired input format accepted by a learner.
- calling the `predict_proba` method of a learner, and manipulating its return values to generate top-N recommendations.

Using the above approach, we develop a recommender system by 'wrapping' the `HoeffdingTree` classifier that is provided by `Scikit-Multiflow`.

Hoeffding Tree (HT) Also known as a Very Fast Decision Tree (VFDT) [65], a HT is an incremental, anytime decision tree inducer designed for learning from data streams. Its key idea lies in the fact that only a small subset of samples passing through a node may be sufficient for deciding on the split attribute. For estimating the minimum number of samples needed, the method employs the Hoeffding bound, which offers sound theoretical guarantees of performance (see [65] for details), asymptotically comparable to that of a batch decision tree.

HT wrapper architecture The `Scikit-Multiflow`'s implementation of a HT supports Naive Bayes prediction at the leaves of the tree, and the possibility of assigning a weight to each fitted sample. We take advantage of both capabilities in the proposed HT recommender. The core idea of our algorithm is to encode all item-to-item associations employed by rule-based methods (see above) in a unified learner. This allows HT to capture both sequential and co-occurrence patterns in a session. The idea is conceptually similar to the BEER[TS] framework [32], but instead of treating between-items associations as separate predictors explored by a bandit, we fit them to a single decision tree classifier using a specific weighting scheme, as explained below. We hence formulate the recommendation task as a multi-class classification problem. Due to the nature of the problem, HT is reduced to a *decision stump*, whose nodes represent input items, and the leaves contain item predictions obtained via Naive Bayes classification.

Training the model The incremental training of a HT in `Scikit-Multiflow` can be done by calling the method `partial_fit(X, y, sample_weight)`. The first two parameters specify the input feature vector and the output label, respectively, with an associated (optional) sample weight. We use these parameters to encode the sequential relation between two items, by letting the feature vector contain the antecedent item and the label represent the consequent item. For the ease of notation, we use the item in place of a feature vector in Algorithm 8 (y or y', lines 7 and 8). Internally, each antecedent is represented as a node of the tree.

Algorithm 8 HT wrapper training procedure

Input: (X, y): sample from the stream; $w_{MC} \geq 1$: importance weight for Markov Chain sequences; $w_{inv} \in [0, 1]$: importance weight for inverse sequences.

```
S ← (y'_1, ..., y'_|S|)        ▷ session item vector for the session id encoded in X
for i ← 1, ..., |S| do
    if i = |S| then
        w ← w_MC               ▷ weight for the 'next-item' sequence
    else
        w ← 1/(|S| − i + 1)    ▷ weight for a non-contiguous sequence
    ht.partial_fit(y'_i, y, w)            ▷ fit observed sequence
    ht.partial_fit(y, y'_i, w · w_inv)    ▷ fit inverse sequence
```

After fetching the current session vector S from the observation window, the wrapper learns all sequential patterns involving item y by fitting a series of samples $(y'_i, y, w), \forall i = 1, \ldots, |S|$, where the sample weight is inversely proportional to the distance between two items. Clearly, these fits utilize the same patterns as captured in Sequential Rules (SR) described above. Among these patterns, $y_{|S|} \rightarrow y$ pertains to the Markov Chain (MC). The corresponding sample, $(y'_{|S|}, y, w_{MC})$, uses a separate weight reflecting the perceived importance of the 'next-item' sequence. Finally, encoding co-occurrence patterns captured by Association Rules (AR) is achieved via a series of the so-called *inverse fits*, i.e. $(y, y'_i, w \cdot w_{inv}), \forall i = 1, \ldots, |S|$, which complete the bidirectional associations. The fixed weight $w_{inv} \in [0, 1]$ is used to inform the influence of inverse (hence unobserved) sequential patterns on the classification.

Making predictions Unlike rule-based models, where the predictions are made solely on the basis of the latest item in a session, our HT wrapper makes predictions in an ensemble-like manner by combining the responses of all the relevant nodes of the tree. The prediction procedure is detailed in Algorithm 9.

For each item y'_i in the current session vector, the wrapper calls the `predict_proba`(y'_i) method of a HT. This method returns a class

Algorithm 9 HT wrapper prediction procedure

Input: X: feature vector containing session id; $w_{MC} \geq 1$: importance weight for Markov Chain sequences; $w_{inv} \in [0, 1]$: importance weight for inverse sequences; N: recommendation cutoff

Output: $\hat{Y}$: top-N recommendations

```
S ← (y'_1, ..., y'_|S|)    ▷ session item vector for the session id encoded in X
P ← (P(y_1), ..., P(y_|P|)), set P(y_i) ← 0, ∀i, ..., |P|    ▷ init probabilities
for i ← 1, ..., |S| do
    P_i ← ht.predict_proba(y'_i)    ▷ predict class probabilities from y'_i
    if i = |S| then
        P_i ← P_i · w_MC
    else
        P_i ← P_i / (|S| − i + 1)
    P ← P + P_i
return Ŷ ← (y_1, ..., y_N), ∀P(y_i) ∈ top-N(P)
```

probability vector that expresses the likelihood of each candidate item to follow item y'_i. The probability vectors P_i are then weighted with the recency of item y'_i. Predictions obtained from the most recent item, $y'_{|S|}$, receive the highest weight specified by w_{MC}. All weighted probability vectors are then added to produce the final scores for the candidate items, top-N of which are returned.

9.4 Simulation results

9.4.1 Datasets

We use public datasets containing sessionized browsing logs. The datasets originate from three recommendation contests representing news, travel, and e-commerce domains. They are summarized in Table 28.

Table 28: Datasets summary (1M events each)

Dataset	*Contest*	*Domain (Action)*	*Items*	*Sessions*	*Avg. length*
Clef	NewsReel'15 [138][33]	News (impressions)	109	305703	3.27 events
Yoochoose	RecSys'15 [21]	E-commerce (clicks)	21300	255166	3.92 events
Trivago	RecSys'19 [143]	Travel (clickouts)	243714	521677	1.92 events

9.4.2 Prequential evaluation setup

We consider the task of online *next-item recommendation*, where the goal is to suggest the list of most probable items to appear in the next session event. We track Recall@10 and MRR@10 during the entire run of the simulation using the real-time visualizer provided by the framework, which reports the global and the current (sliding) averages. Time horizon for each simulation is set to 1M events, with evaluation and observation window sizes of 10K and 50K events, respectively. The latter size was chosen in consideration of sufficient (sliding) session history and a reasonable memory/runtime overhead. We use a separate validation set of 100K events (preceding those of the simulation) for hyperparameter tuning. The evaluation itself is performed from pure cold-start, with no model pre-training involved. Sessions of size 1 are excluded from the evaluation. For our simulations, we use Intel Core i7 CPU @ 2.80 GHz and 16Gb RAM.

9.4.3 Model setup

We evaluate the models presented in Section 9.3, with an addition of a Random classifier that sets the lower bound for performance. Below we briefly outline the optimal model configurations after the hyperparameter tuning step.

Rule-based (AR, MC, SR) models operate on a global scale, whereas the popularity-based one (POP) works within a sliding window. S-kNN uses Cosine similarity, with $k = 100$ (Trivago), $k = 200$ (Clef), $k = 300$ (Yoochoose). We also use *recent session sub-sampling* [120], with sub-sample sizes of 500 (Clef), 1000 (Trivago), and 1500 (Yoochoose). BEER[TS] includes AR, MC, SR, POP, and S-kNN as components of the ensemble. The HT wrapper uses weights $w_{MC} = 3$ (Yoochoose), $w_{MC} = 5$ (Clef, Trivago), and $w_{inv} = 0.01$ (Clef), $w_{inv} = 0.9$ (Yoochoose, Trivago). The `HoeffdingTree` class of `Scikit-Multiflow` is instantiated with `leaf_prediction='nb'` to enable Naive Bayes prediction at the leaves. The pre-configured experiments are provided in `FlowRec`'s code base for reproducibility.

[33] We use the subset of the dataset provided in [173].

9.4.4 Results

The results of the simulation for the three datasets are presented in Figures 22-24. The top performing model in each case is marked with an asterisk. Note that the column 'Sliding' contains the averages of the *very last* evaluation window.

The HT wrapper consistently achieves higher recall and MRR than the other baselines on all three datasets. This proves the effectiveness of combining item-to-item sequential patterns, and utilizing the entire user session at prediction time in an ensemble-like manner. The HT wrapper also happens to be noticeably faster than its main rival, BEER[TS] (in its current configuration). The total running times for each model are recorded by the framework. As a point of reference, we consider the upper limit of 100 ms per recommendation prescribed by the CLEF NewsReel challenge [138]. Table 29 reports average response times per recommendation request for each model. We observe that all runtimes fall within the recommended limit.

The real-time visualization offers the possibility to diagnose potential issues at an early stage. For instance, the performance charts for Yoochoose and Trivago make it obvious that the inclusion of POP to BEER[TS] is not justified, and hence it can be dropped from the ensemble to gain speed. The live monitoring of model evolution helps to see how algorithms behave relative to each other on various segments of the dataset, as well as to better understand the peculiarities of the dataset itself. For example, the above two charts clearly show the stagnation of S-kNN after leaving the initial cold-start segment ($\approx$0-50K), while other models continue to learn. We also see that Clef exhibits a more dynamic domain (news) with more profound concept drift, which makes learning more challenging. It is the only dataset where the popularity recommender has decent performance, surpassing other models on certain data segments. The Clef chart also reveals the 'easy' portion of the dataset ($\approx$ 600K-800K), where most algorithms (but not POP) boost their performance.

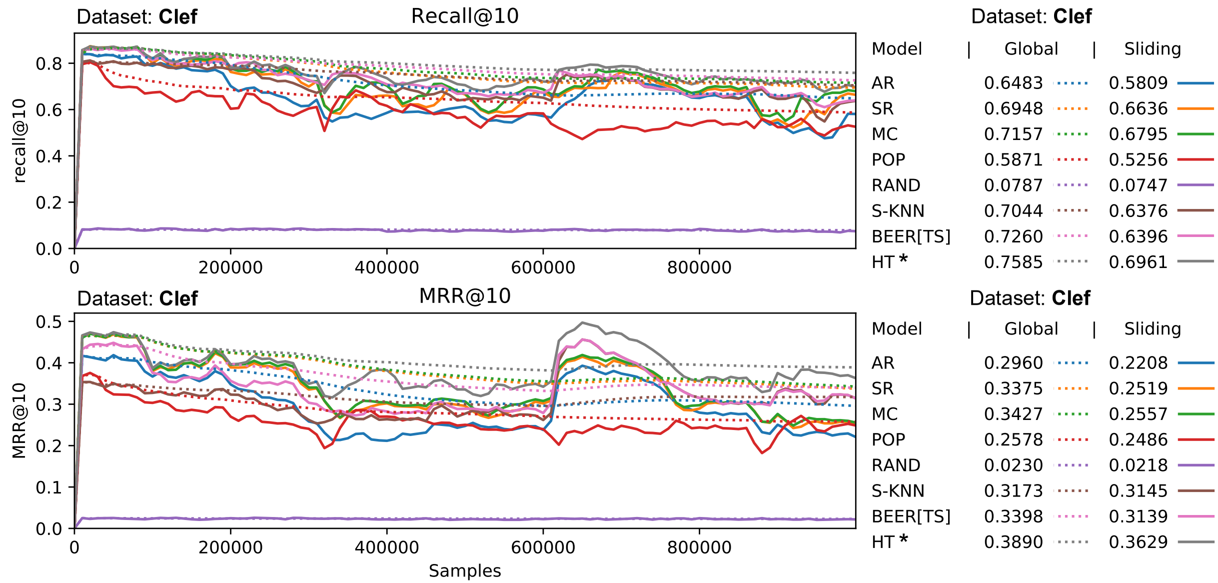


Figure 22: Performance charts for Clef (news)

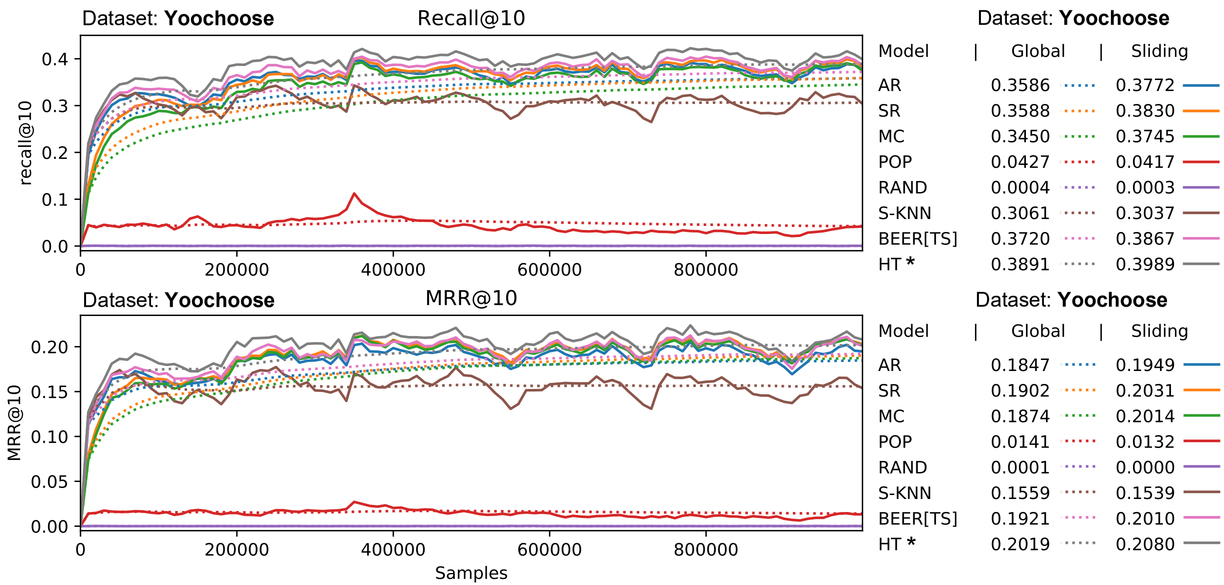


Figure 23: Performance charts for Yoochoose (e-commerce)

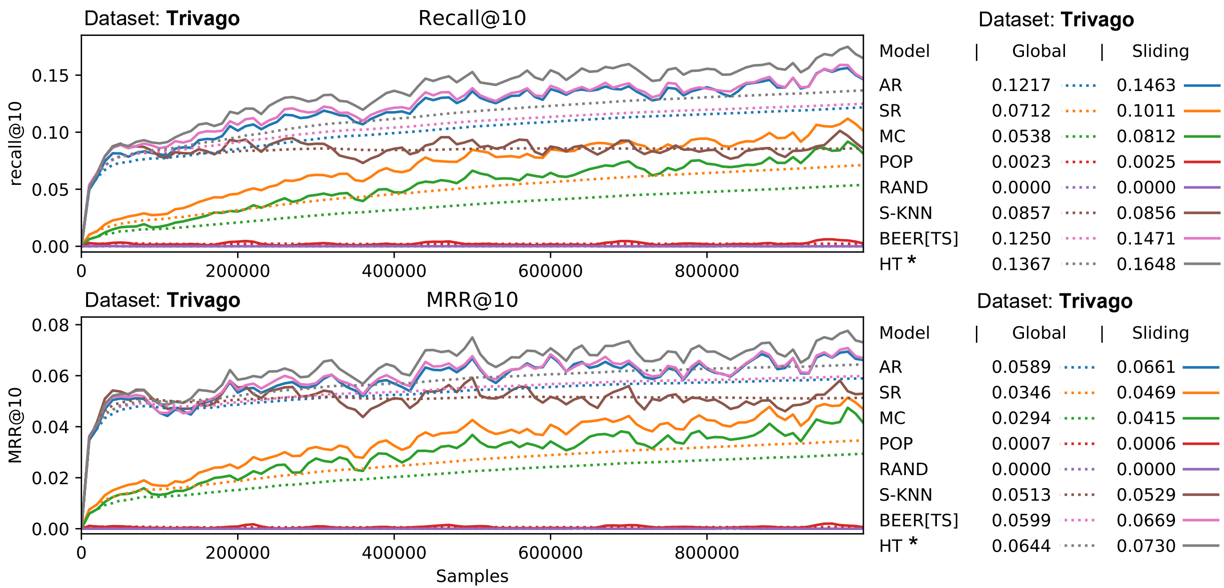


Figure 24: Performance charts for Trivago (travel)

Table 29: Average recommendation time (msec) per model

	AR	*SR*	*MC*	*POP*	*RAND*	*S-kNN*	*BEER[TS]*	*HT*
Clef	0.270	0.213	0.204	0.187	0.204	26.543	27.858	2.542
Yoochoose	0.744	0.623	0.476	3.365	0.313	2.434	13.331	9.629
Trivago	4.276	4.093	4.082	11.38	1.150	5.675	85.163	23.036

9.5 Conclusion

We introduce `FlowRec` — a new recommendation framework for streaming session data developed on top of `Scikit-Multiflow`. It serves as a testbed for streaming recommendation models by offering prequential evaluation with real-time performance monitoring. One advantage of prototyping in `FlowRec` is the ability to 'wrap' various stream learners provided by `Scikit-Multiflow`, thus treating them as black boxes. We demonstrate how to develop such a wrapper for the `HoeffdingTree` class, capable of generating accurate session-based recommendations on evolving data streams. The framework will be further extended with additional evaluation protocols, metrics, and algorithms.

10 PAPER V

Paraschakis D. and Nilsson B.J.

"Matchmaking Under Fairness Constraints: A Speed Dating Case Study"

Abstract Reported evidence of biased matchmaking calls into question the ethicality of recommendations generated by a machine learning algorithm. In the context of dating services, the failure of an automated matchmaker to respect the user's expressed sensitive preferences (racial, religious, etc.) may lead to biased decisions perceived by users as unfair. To address the issue, we introduce the notion of *preferential fairness*, and propose two algorithmic approaches for re-ranking the recommendations under preferential fairness constraints. Our experimental results demonstrate that the state of fairness can be reached with minimal accuracy compromises for both binary and non-binary attributes.

Published in International Workshop on Algorithmic Bias in Search and Recommendation (Bias 2020). Held as part of the 42nd European Conference on Information Retrieval (ECIR 2020).

10.1 Introduction

In 2016, a number of incidents were reported by users of the dating app CoffeeMeetsBagel[34], who complained that the service had consistently been matching them against partners of their own ethnicity, despite the user's explicitly communicated ethnic neutrality [210]. Arguably, the observed bias of algorithmic matchmaking originates from two common assumptions: first, that an algorithm must strive to reproduce the preferences of existing users; and second, that these *inferred* preferences should inform matching decisions, overriding the user's *expressed* desires [117]. Another motivating example [293] shows that a classifier trained on speed dating data can learn to discriminate on the basis of protected characteristics of users, unless special preventive measures are taken. One such measure is re-ranking the recommendation output in a post-processing step, which can be applied to traditional recommender systems trained in pointwise or pairwise manner [273]. The idea of *preferentially fair* matchmaking therefore boils down to formulating and solving a recommendation problem under expressed preference constraints. In line with the above examples, we investigate the fairness of a matchmaking mechanism for speed dating, based on a publicly available dataset [79].

10.2 Case study

We begin our study by analysing how consistent people are in following their intimate preferences. The aforementioned dataset contains 4189 speed dates collected over a series of 21 meetups (a.k.a. 'waves') between 2002 and 2004. The participants were Columbia University students who all had a 4-minute date with each person of the opposite sex. At the end of each date, both parties had to make (or not make) their pick. In case of reciprocal liking, a 'match' was registered and the contact details were exchanged. Before attending, they filled in a pre-registration questionnaire to state their demographics, self-perceived traits, and preferences. In particular, attendees could express how important it was for them to date people of their own race or religion.

[34] **https://coffeemeetsbagel.com/**

In Figure 25, we compare the distributions of own-race and other-race partners in the candidate pool with the corresponding distributions of eventually liked partners (referred to as 'picks' in Figure 25). We observe that for low values of own-race importance (namely, 1-4), the racial distribution of picked partners closely follows that of the candidate pool. The pattern starts changing after the value of 7 onwards, where we notice discrepancies between the racial distribution of candidate partners and the picked ones; namely, showing far less interest in other-race partners. This proves that people generally tend to follow their racial preferences, which therefore should be respected by a matchmaker.

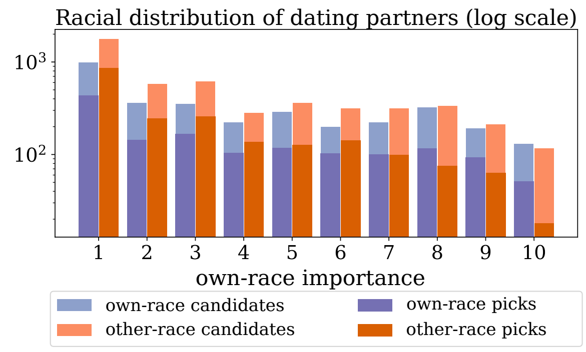


Figure 25: Consistency patterns in racial preferences

10.3 Matchmaking

A matchmaker designed for speed dating offers assistance in mate selection and helps participants find their matches quicker. It works by comparing the content-based profile of a target user with the corresponding profiles of all other attendees of the opposite sex within the same wave, in attempt to predict a pick (individual positive decision) or a match (reciprocal positive decision). It is trained on historical data from the remaining waves containing the outcomes of all their dates. This describes a hybrid recommendation model known as *collaboration via content* [224], which practically boils down to a binary classification task. The percentage of picks and matches in the dataset are 42% and

16%, respectively. To tackle class imbalance, we employ cost-sensitive learning and classification threshold tuning established through cross-validation on training data.

The dataset structure offers a natural way of performing a 21-fold cross-validation, where each wave is used once as a test set, while the remaining waves are used for training. We use the aggregated *F1 score* as our accuracy metric:

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{42}$$

The number of participants in a wave varies from 9 to 44. Since each target user appears only in his/her wave, the matchmaker cannot rely on the their prior dating history and hence operates in the conditions of continuous cold-start.

Due to small candidate pools, the purpose of a matchmaker in speed dating is to suggest *all* good candidates to a user (as opposed to the classical top-k recommendation). In other words, it must decide not only *who* to recommend, but also *how many* of them. The cut-off k is thus dynamically determined by the classification threshold. This 'variable-k' recommendation setting can be generalized to other domains beyond speed dating that operate on short candidate lists (e.g. suggesting restaurants in the immediate neighborhood of a user).

10.4 Related work

Speed dating is a relatively new format, which has become the topic of several studies [79, 80, 123, 293, 327] aiming to establish the key influential factors of 'rapid' matchmaking. First insights were provided by Fisman et al. [79, 80], who showed that women tend to put much emphasis on the partner's race and ethnicity. Van der Zon [293] builds a classifier for speed dating to predict the partner's positive decision, and shows that it can produce racially discriminatory rules. Issues of bias and discrimination on intimate platforms have been studied by Hutson et al. [117], who advocate the design of matchmaking services able to intervene in discrimination without overriding the user's preferential autonomy. All the above insights motivate our study as they stress the need for the careful consideration of sensitive attributes in a non-invasive way when making mate suggestions.

A reciprocal recommender for speed dating was recently proposed by Zheng et al. [327]. The authors focus on the multi-stakeholder notion of fairness, which in their case does not involve any protected attributes and hence is less ethically charged than the interpretation of fairness that we put forward in this paper. Joel et al. [123] investigate whether the mutual attraction in speed dating can be predicted from the users' traits and preferences reported before the date. This is identical to the experimental setup of our work. Despite the use of cutting-edge machine learning algorithms, predicting a match above chance has proved to be extremely challenging [123]. Although the predictive accuracy is not the focal point of our study, our experience with predicting matches has been similar. Sapiezynski et al. [254] propose a new fairness metric for ranked outputs, in which fairness depends not only on the ranking algorithm itself, but also on the model of user attention. This way, the same ranking can appear biased both in favor and against the protected group depending on how the user attention is distributed over items. This metric is used to reveal societal biases in online dating, which can be corrected by reshuffling the system's outputs. Although different in approach, the general idea of their work is similar to ours.

The role of fairness-aware algorithms in promoting non-discrimination, neutrality, and fair treatment in recommender systems has gained wide recognition [34, 130, 132, 168, 310, 314, 328]. Fairness is a multi-faceted notion allowing a range of definitions, e.g. see [84, 294]. It is also a multisided concept [34] as it concerns both parties: those receiving recommendations, and those being recommended. This obviously holds true in dating contexts. Steck [273] expresses fairness in terms of preserving the proportionality of user interests in recommendations (a.k.a. 'calibration'). Conversely, Kamishima et al. [130] define fairness as staying neutral to user's viewpoints to enforce diversity in recommendations and avoid filter bubbles. In general, diversity and fairness are closely related concepts [132, 157, 168].

Our work is along the lines of recent studies focusing on how to achieve fairness via re-ranking [24, 132, 168, 313, 319], and how to quantify the fairness of existing rankings [313, 314]. Adopting this approach typically incurs tolerable compromises in prediction accuracy [168, 273]. We empirically validate this hypothesis in Section 10.7.

10.5 Preferential fairness

10.5.1 Background

Presently, at least 20 different definitions of fairness are known in the machine learning community [84, 294]. In a broad sense, preferential fairness falls under the category of *individual fairness*, which requires that 'similar individuals are treated similarly' [68]. To relate it to our case, two individuals can be considered similar if they have expressed similar racial preferences. All other things being equal, the 'treated similarly' part implies that they both receive partner recommendations with similar racial distributions.

Two edge cases are possible. When the strongest preference is expressed, the matchmaker is restricted to recommendations of own-race partners to satisfy the user's request. From the ethical viewpoint, acting differently would mean a violation of the freedom of choice (i.e. depriving the user of the ability to form a relationship with a partner of the desired race). Conversely, the weakest preference implies that the user is equally interested in all the races. Ignoring this preference could lead to unjustified racial biases akin to filter bubbles [130]; see CoffeeMeetsBagel example above [210].

In most cases, the distribution of races (or any other sensitive attribute) is not uniform in the candidate pool, which raises the question of how to sample the candidates. Should all the races have equal representation in recommendations, or should it be proportional to the racial distribution of the candidate pool? A recent user study investigating public attitudes towards fairness [259] shows that the latter option is generally perceived as the fairer choice in such scenarios[35]. Selecting individuals in proportion to their merit is known as

[35] The assumption of proportionality may not hold for all individuals. More user studies are needed to better understand what users actually mean by 'no racial preference'.

calibrated fairness, whose several interpretations exist [169, 273]. Its general idea is rooted in the theory of *proportional equality* conceived by the ancient philosopher Aristotle, and serving as a basis for distributive justice [92]. Therefore, preferential fairness is a special case of calibrated fairness, where the calibration is done on the basis of expressed user preferences. We argue that this meritocratic formulation also satisfies the conditions for *multi-sided fairness* [34], where the ethical treatment of both parties (i.e. users and candidates) is taken into account. As follows from Figure 25, it is in the interest of both sides that candidates are selected in proportion to the target user's expressed preference for race because it increases the chance of a positive outcome. In practice, the merit for picking a candidate for recommendation is a combination of his/her level of compliance with the target user's sensitive preferences, and the estimated probability of their match. In the meritocratic sense, this serves as a 'minimal guarantee of fairness' [125]: any candidate who is presently more qualified than another candidate should have a better (or at least equal) chance of exposure as the less qualified candidate. Thus, the task of a preferentially fair matchmaker is to calibrate its recommendations by mapping the user's preference to the distribution of a sensitive attribute in the candidate pool.

10.5.2 Model

To make calibration possible, we need to find the optimal mean μ_u^* for the race attribute of the generated k-sized recommendation list for user u. The mean μ_u^* should reflect the distribution of this attribute in the candidate pool in proportion to the user's expressed preference for race. For our running example, let us consider a binary sensitive attribute $a \in \{0, 1\}$, and the user's associated degree of preference for this attribute, $p_u \in [0, 1]$. Let $A_C = (a_1, a_2, \cdots, a_n)$ denote the attribute distribution of the complete candidate pool C, and define the two values μ_{max} and μ_{A_c} to be the mean of the k largest attribute values in A_C and the mean of all the n attribute values in A_C, respectively. According to the previously defined idea of fairness, we can define the optimal mean as:

$$\mu_u^* = (1 - p_u) \cdot \mu_{A_C} + p_u \cdot \mu_{max} \tag{43}$$

Further, let $A_u = (a_1, a_2, \cdots, a_k), k < n$ be the attribute distribution of the recommendation list for user u. By analogy, μ_{A_u} is the mean of the attribute values in A_u.

If we encode the race attribute for user u such that the value of 0 denotes 'other race' and the value of 1 denotes 'own race', it is easy to see that the above equation satisfies both edge cases presented earlier. For $p_u = 1$ (strong preference), the optimal mean enforces the maximal skewness of recommendations towards own-race partners, whereas for $p_u = 0$ (weak preference), the optimal mean reduces to the mean of the candidate pool.

Therefore, the calibrated fairness of a recommendation list can be expressed by its closeness to the optimal mean. To be able to quantify this fairness on a $[0, 1]$ scale, we first compute its offset from the optimal mean, $\delta_u = |\mu_u^* - \mu_{A_u}|$. We then find the minimum and the maximum offsets $\delta_{min} = \min_\mu |\mu_u^* - \mu|$ and $\delta_{max} = \max_\mu |\mu_u^* - \mu|$, where the means μ are taken over all possible item combinations of size k from the candidate pool, C. This allows us to formally define the *preferential fairness* ϕ_u of a given user's recommendation list as follows:

$$\phi_u \overset{\text{def}}{=} 1 - \frac{\delta_u - \delta_{min}}{\delta_{max} - \delta_{min}}, \tag{44}$$

where the term $(\delta_u - \delta_{min})/(\delta_{max} - \delta_{min})$ represents the *unfairness penalty* henceforth denoted by ε_u.

In the extreme case when all candidates share the same attribute value (e.g. they are all of the same race), we simply set $\phi_u = 1$ to avoid division by zero. ϕ_u takes values from 0 to 1, where higher values suggest greater fairness and we can also express it as a percentage score.

10.6 Re-ranking methods

The above measure allows us to quantify the fairness of recommendations with respect to the user's sensitive preferences. Further, it can be used to constrain a potentially biased recommender system in order to make it fair(er). As mentioned earlier, the recommender system is essentially a binary classifier which outputs a vector of matching scores for all the candidates, and recommends the ones above some threshold value, θ. The generated predictions can be optimized for fairness using re-ranking methods, with the possibility to control the accuracy-fairness trade-off via a tolerance constant $\alpha \in [0, 1]$. We refer to the resulting solutions as α-fair recommendations, whose fairness is bounded by the condition $\varepsilon_u \leq \alpha$.

10.6.1 Knapsack

One way of modeling the accuracy-fairness trade-off is by viewing it as a variant of a *0-1 Knapsack problem* (KP). In its classical formulation, we are given a set of items $\{x_i\}_{i=1}^{n}$ each with a weight $w_i > 0$ and a value $v_i > 0$, and are tasked with filling the knapsack with items that would maximize its total value without exceeding its upper bound on the weight capacity W:

$$\begin{aligned} \max \sum_{i=1}^{n} v_i x_i \\ \text{subject to} \sum_{i=1}^{n} w_i x_i \leq W, \text{and } x_i \in \{0, 1\} \end{aligned} \tag{45}$$

The Knapsack problem is NP-complete, and can be solved to optimality by means of dynamic programming (DP) in pseudo-polynomial time [180].

Equation 45 clearly resembles our task of maximizing the overall accuracy of recommendations (i.e. the sum of prediction scores) without exceeding the unfairness score ε_u bounded by α. Let A_C and S_C denote the attribute vector and the prediction score vector of the candidate itemset, respectively. Knapsack weights and values can then be represented as $w_i = A_C[i]$ and $v_i = S_C[i]$, for $1 \leq i \leq n$, where $n = |C|$ is the total number of candidates. Because the resulting capacity of our knapsack (i.e. the sum of attribute values) must be evaluated in terms of its (un)fairness, two important adaptations are needed.

Adaptation 1: lower bound on capacity. The tolerance constant α basically defines the allowed range of means for the attribute vector of a candidate solution. Hence, there is a need to satisfy not only the upper bound (as in the classical KP), but also the *lower bound* on the knapsack capacity in the DP algorithm, i.e. $\sum_{i=1}^{n} w_i x_i \geq W_L$. This variant of the Knapsack problem has rarely been addressed in the literature (we are only aware of two works, namely [41, 312]).

To compute the bounds satisfying the given tolerance α, we first find the corresponding δ_α offset:

$$\delta_\alpha = (1-\alpha) \cdot \delta_{min} + \alpha \cdot \delta_{max} \tag{46}$$

Let $\mu_L = \max(\mu_u^* - \delta_\alpha, \mu_{min})$ and $\mu_U = \min(\mu_u^* + \delta_\alpha, \mu_{max})$. Then any α-fair solution R^α must have its attribute mean in the interval $\mu_L \leq \mu_{A_{R^\alpha}} \leq \mu_U$.

Considering that $\mu_{A_{R^\alpha}} = \frac{\sum_{w \in A_{R^\alpha}} w}{|R^\alpha|}$, we get:

$$W_L \overset{\text{def}}{=} |R^\alpha| \cdot \mu_L \leq \sum_{w \in A_{R^\alpha}} w \leq |R^\alpha| \cdot \mu_U \overset{\text{def}}{=} W_U \tag{47}$$

The above equation defines the bounds on the knapsack capacity guaranteeing α-fairness at size $k = |R^\alpha|$.

Adaptation 2: exact-k solution The original KP formulation does not restrict the size of a solution. Since the computation of fairness is done for some specific list size k, we introduce a new constraint: $\sum_{i=1}^{n} x_i = k$. This variant is known as the *exact k-item Knapsack problem* (E-kKP) [42], and can be solved by adding a 3rd dimension to the DP algorithm to keep track of the knapsack size.

Since our end goal is to find the most accurate ranking of arbitrary size such that it satisfies the fairness constraint, we run a separate instance of E-kKP for each $k = 1, \ldots, n-1$ to obtain a set of α-fair solutions $\mathcal{F} = \{R_k^\alpha\}_{k=1}^{n-1}$, where R_k^α is a proper subset of C. Note that the DP algorithm simply optimizes the sum of item values without distinguishing between positive ($v_i \geq \theta$) and negative ($v_i < \theta$) predictions. Since KP does not support negative values, we introduce a *relevance* *criterion* that allows us to pick the optimal solution R^* as follows:

$$R^* = \arg\max_{R^\alpha \in \mathcal{F}} \frac{\min(|R|, |R^\alpha|)}{\max(|R|, |R^\alpha|)} \cdot \left(\sum_{v \in S_{R^\alpha}} v - \theta \right) \tag{48}$$

where R is the original (presumably unfair) recommendation list, and S_{R^α} is the prediction score vector of an α-fair solution, R^α.

The first factor in Equation 48 optimizes the F1 score by penalizing solutions whose sizes diverge from the original recommendation list (causing losses in precision or recall), whereas the second one penalizes solutions containing negative predictions. The criterion is generalizable to any re-ranking method, and can potentially be adjusted to optimize other measures of interest.

Adapted KP The above adaptations are unified in a KP variant with cardinality and dual capacity constraints:

$$\begin{gathered} \max \sum_{i=1}^{n} v_i x_i \\ \text{subject to } W_L \leq \sum_{i=1}^{n} w_i x_i \leq W_U, \sum_{i=1}^{n} x_i = k, x_i \in \{0,1\} \end{gathered} \tag{49}$$

A DP solution can be constructed based on the following recurrence relation:

$$T[i,k,l,u] = \max \begin{cases} T[i-1,k,l,u], \\ \text{if} \sum_{w \in \{k \text{ largest of } \{w_j\}_{j=i}^{n}\}} w \geq l \\ \\ T[i-1,k-1,l-w_i,u-w_i] + v_i, \\ \text{if} \sum_{w \in \{k-1 \text{ largest of } \{w_j\}_{j=i}^{n}\}} w \geq l - w_i, w_i \leq u \end{cases} \tag{50}$$

where indices i, k, l, u correspond, respectively, to item index, solution size, capacity lower bound, capacity upper bound.

Since for binary attributes the upper bound is at most n, the overall time complexity of the KP re-ranking algorithm is $\mathcal{O}(n^4)$, yielding an optimal solution that would otherwise require $\mathcal{O}(n \cdot 2^n)$ time to compute the score in Equation 48 in the case of an exhaustive search.

It turns out that a special case of KP can also be used to speed up the computation of the fairness measure. Recall that it relies on quantities δ_{min} and δ_{max}, where the former is expensive to compute as it requires enumerating all combinations. An alternative solution is to formulate it as a *subset sum* problem [6], which is a variant of KP with item values being equal to their weights. The goal is to determine if some subset of items sums up to a predefined number t. In our case, $t = k\mu^*$. But instead of searching for exact matches, we seek a subset whose sum of values is *the closest* to t. Again, we use DP to fill in the memoization array $T[n, k, W_u]$, from which it is easy to compute δ_{min}:

$$\delta_{min} = \arg\min_{j=1,\dots,W_U} \frac{|t - T[n,k,j]|}{k} \tag{51}$$

Continuous attributes To use KP with continuous attributes, we simply multiply the fractional weights by a scaling factor and take the integer part. Clearly, the choice of a scaling factor will affect the precision of the solution and the running time. This would allow us to work with attributes that contain, for instance, probabilities of candidates being of the same religion as the target user.

10.6.2 Tabu search

Another re-ranking method we develop is a *hill climbing* heuristic based on *Tabu search*. It starts with an initial solution and gradually improves it according to some optimization criterion until the stopping condition is met. Unlike Knapsack, Tabu search does not guarantee an optimal solution, but has the advantage of finding a good approximation fast[36].

Here, the initial solution is the original (presumably unfair) recommendation list R. In each iteration, the algorithm chooses between two types of operations:

1. Adding the highest-scored item from $C \setminus R$ to R, yielding the solution R^+
2. Removing the lowest-scored item from R, yielding the solution R^-

[36] Due to the termination criterion, the precise running time of Tabu search is unknown

To drive these choices, the algorithm evaluates both possibilities using the criterion given in Equation 48 with $\mathcal{F} = \{R^+, R^-\}$, and selects the maximum one. The 'tabu' part restricts the neighborhood of candidates to those that:

1. Will move the resulting mean in the direction of μ_u^*
2. Have not been used in the previous iteration(s)

This is implemented by maintaining a dynamic tabu list containing currently prohibited items. The algorithm terminates as soon as the unfairness penalty ε_u of the solution becomes lower or equal to the chosen tolerance α.

Continuous attributes Escaping from local minima can be challenging for Tabu search in case of continuous attributes. We solve this by employing a restarting mechanism that is triggered when a limit of iterations has been reached. With each algorithm restart, the size of the tabu list is incremented by 1.

10.7 Experimental results

We implement a matchmaker for speed dating[37] as described in Section 10.3, using four different classifiers: Logistic Regression, Random Forest, XGBoost, and Gaussian Naive Bayes. The target label is 'match' (i.e. reciprocal liking). Our implementation follows the standard pipeline involving missing data imputation, feature selection/encoding, and hyperparameter tuning. After the pre-processing step, there are 39 total attributes describing each pair of potential dating partners. We measure the F1 score and the fairness (in %) of generated recommendations before and after re-ranking. We set $\alpha = 0$ to enforce 100% fair solutions.

10.7.1 Racial bias

We first resolve the *racial* bias of recommendations on the basis of two attributes: (a) the 'same-race importance' attribute expressed by the

[37] Source code: **https://git.io/preferential_fairness**

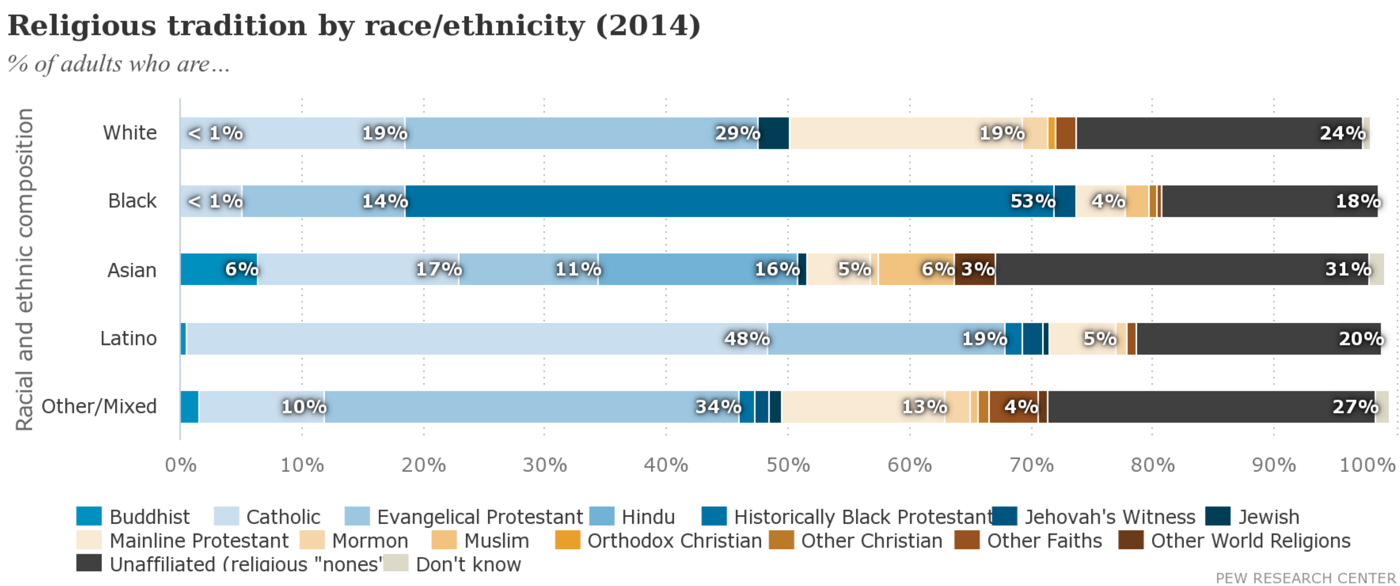


Figure 26: Religious tradition by race/ethnicity (227)

user on an ordinal scale from 1 to 10, and (b) the binary 'same race' attribute indicating whether the given pair belongs to the same race (in essence, this attribute is a result of collapsing a complex multivalued 'race' variable into a single binary one)[38]. For re-ranking, we employ the two methods introduced in the previous section. The results are summarized in Table 30.

10.7.2 Religious bias

Resolving the *religious* bias requires a slightly more involved procedure. For reasons beyond our knowledge, the speed dating dataset only provides the 'importance of the same religion' preferential attribute, without the corresponding 'same religion' attribute on which re-ranking methods could rely. One way to address the problem is to estimate the missing attribute probabilistically from external data sources. Although possibly lacking in precision, such estimates are commonly acceptable for illustration purposes (e.g. inferring race from profile pictures, as done in [254]). We make our estimation of religion on the basis of the U.S. religious landscape study [227]. It contains statistics about the racial and ethnic composition by religious group, covering all 50 U.S. states[39]; see Figure 26. Let $X = \{x_1, x_2, \cdots, x_n\}$ be the set of n religions and $Y = \{y_1, y_2, \cdots, y_m\}$ be the set of m races. Given the knowledge about the races of users, the probability of two users u_1 and u_2 being of the same religion can be estimated as follows: $\mathbb{P}(x_{u_1} = x_{u_2} | y_{u_1}, y_{u_2}) = \sum_{x \in X} \mathbb{P}(x|y_{u_1}) \cdot \mathbb{P}(x|y_{u_2})$.

After the analysis, the minimum and the maximum estimates correspond, respectively, to combinations $\langle Black, Asian \rangle$ with $\mathbb{P} = 0.09$, and $\langle Black, Black \rangle$ with $\mathbb{P} = 0.34$.

To use re-ranking methods with probabilistic attributes, we apply the methodologies described in the previous section. For Knapsack, we multiply all weights by a scaling factor of 10^3 to keep good balance between running time and precision of a DP algorithm. For Tabu

[38] A limitation of this approach is that it cannot properly match participants having biracial or multiracial identity

[39] Although the speed dating study also took place in the U.S., we allow some margin of error due to state-to-state variability of racial/religious composition

Table 30: Re-ranking under *racial* fairness constraints

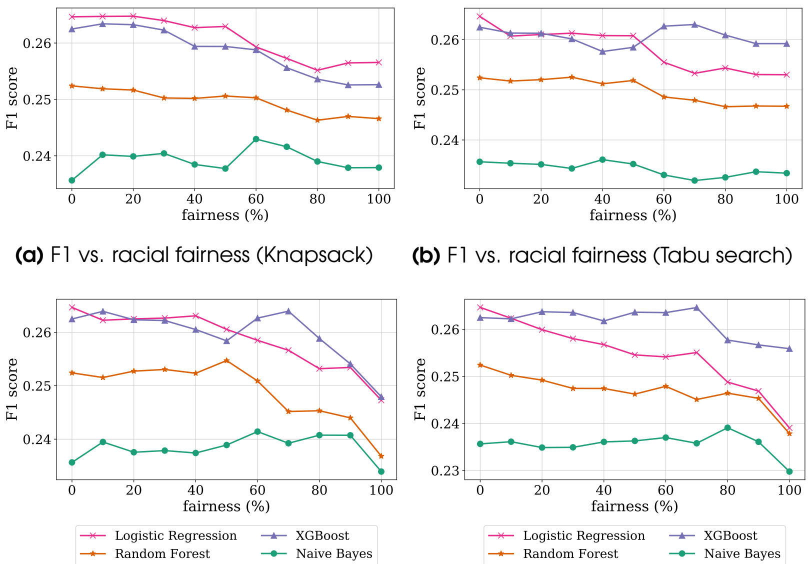
Measure	Original	Fair (Knapsack)	Fair (Tabu)
	Logistic Regression		
F1 score	0.2647	0.2566	0.2530
Fairness	54.71%	100%	100%
	Random Forest		
F1 score	0.2524	0.2466	0.2467
Fairness	53.28%	100%	100%
	XGBoost		
F1 score	0.2625	0.2526	0.2592
Fairness	51.99%	100%	100%
	Gaussian Naive Bayes		
F1 score	0.2357	0.2379	0.2334
Fairness	52.63%	100%	100%

Table 31: Re-ranking under *religious* fairness constraints

Measure	Original	Fair (Knapsack)	Fair (Tabu)
	Logistic Regression		
F1 score	0.2647	0.2473	0.2390
Fairness	56.31%	100%	100%
	Random Forest		
F1 score	0.2524	0.2368	0.2378
Fairness	52.31%	100%	100%
	XGBoost		
F1 score	0.2625	0.2479	0.2559
Fairness	49.69%	100%	100%
	Gaussian Naive Bayes		
F1 score	0.2357	0.2339	0.2298
Fairness	58.63%	100%	100%

search, we employ a restarting mechanism to escape from local optima. We set the initial tabu list size to 2, and the limit of iterations to 100. According to our observations, it can take up to 5 algorithm restarts to obtain a solution. However, occasions like these have been very rare. The results of re-ranking with religious preferences are summarized in Table 31.

To better see the impact of re-ranking on accuracy at various fairness levels, we compare their trade-off for racial and religious preferences in Figure 27.



(a) F1 vs. racial fairness (Knapsack) **(b)** F1 vs. racial fairness (Tabu search)

(c) F1 vs. religious fairness (Knapsack) **(d)** F1 vs. religious fairness (Tabu search)

Figure 27: F1 score vs. fairness

10.7.3 Discussion

It appears that the preferential fairness of off-the-shelf classifiers before re-ranking does not exceed 55% for race, and 59% for religion (column 'Original' in Tables 30 and 31). Nevertheless, maximizing fairness from the state of being nearly 'half fair' in the beginning is possible with minimal accuracy losses thanks to the proposed re-ranking methods.

As expected, the losses in F1 observed in Table 31 are more substantial than those in Table 30, reflecting a more demanding nature of re-ranking over non-binary attributes. Specifically, the largest registered accuracy drop is 9.7% for religion, and 4.4% for race.

During the experiments, we observe that re-ranking of a non-oracle recommender occasionally triggers true positives or true negatives. The more accurate the recommender, the higher the loss due to re-ranking. Indeed, the largest decrease in F1 score is associated with our most accurate classifier, Logistic Regression. Conversely, our least accurate classifier — Naive Bayes — actually enjoys a small increase in F1 after re-ranking via Knapsack (Table 30).

The trajectory of F1 loss for increasing fairness levels differs from model to model (Figure 27). In general, we see a steeper decline in accuracy for probabilistic attributes (Figures 27c, 27d), especially as the fairness approaches 100%.

The two re-ranking methods have comparable performance in practice, without a clear winner. The choice of a method should be dictated by the nature of a sensitive attribute (whether it is binary or continuous), and the size of the candidate pool. Knapsack has the advantage of guaranteed optimality for binary attributes, and potentially better runtime efficiency for continuous attributes (if a reasonable scaling factor is chosen). On the other hand, Tabu search has higher efficiency for larger candidate pools.

In view of the modest accuracy compromises, and of the legal and ethical ramifications of racially and religiously biased predictions, the use of re-ranking is easily justifiable.

10.8 Conclusion

In response to the provided evidence for algorithmically biased matchmaking, we introduce the notion of *preferential fairness* – a special case of calibrated fairness, where the user's preference for the sensitive attribute and its distribution in the candidate pool set the merit for choosing the right candidates for recommendation. Using the proposed fairness measure and re-ranking algorithms, we show how to quantify and eliminate racial and religious bias in the outputs of matchmaking classifiers for speed dating. This measure can be useful in other domains where the preferential aspect of fairness is the key factor for establishing the ethicality of recommendations.

As a theoretical contribution, we have formulated and solved the exact-k Knapsack problem with dual capacity constraints, which guarantees optimality in re-ranking. We also present a variation of Tabu search, offering potentially better scalability and comparable re-ranking performance. Both methods are driven by the proposed relevance criterion, which maximizes the accuracy of fair recommendations to a user. Contrary to the established 'top-k' standard, we address a harder problem of generating recommendations with a dynamic cut-off. Our methodology can be generalized to other scenarios where multi-objective, variable-size recommendations are meaningful. Re-ranking under multiple fairness constraints would be an interesting avenue for future work.

To conclude, we agree with the vision that future designs and policies of matchmaking services should strive to discourage users from expressing socially sensitive preferences to protect the dignity and self-esteem of the concerned minority groups [117]. We show that when such preferences do exist, the proposed methodology can aid in assessing and de-biasing the output of a recommender system for the affected user.

REFERENCES

[1] H. Abdollahpouri and R. Burke. Multi-stakeholder recommendation and its connection to multi-sided fairness. In *Proceedings of the Workshop on Recommendation in Multi-stakeholder Environments co-located with the 13th ACM Conference on Recommender Systems (RecSys 2019)*, volume 2440 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2019.

[2] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 217–253. Springer US, 2011.

[3] D. Agarwal, B.-C. Chen, P. Elango, and R. Ramakrishnan. Content recommendation on web portals. *Commun. ACM*, 56(6):92–101, June 2013.

[4] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, 1993.

[5] F. Aiolli. Efficient top-n recommendation for very large scale binary rated datasets. In *Proc. of the 7th ACM Conference on Recommender Systems*, pages 273–280, 2013.

[6] R. J. Alfonsín. On variations of the subset sum problem. *Discrete Applied Mathematics*, 81(1-3):1–7, 1998.

[7] Z. Ali, S. Khusro, and I. Ullah. A hybrid book recommender system based on table of contents (toc) and association rule mining. In *Proc. of the 10th International Conference on Informatics and Systems*, INFOS '16, pages 68–74. ACM, 2016.

[8] H. Allcott and M. Gentzkow. Social media and fake news in the 2016 election. *NBER Working Paper No. 23089*, 2017.

[9] X. Amatriain and J. Basilico. Netflix recommendations: Beyond the 5 stars. **http://techblog.netflix.com/2012/04/netflix-recommendations-beyond-5-stars.html**, Apr. 2012. Accessed: 2015-03-12.

[10] X. Amatriain, A. Jaimes, N. Oliver, and J. M. Pujol. Data mining methods for recommender systems. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 39–71. Springer US, 2011.

[11] E. Andrews. Recommender systems for online dating. Master's thesis, University of Helsinki, 2015.

[12] S. C. Anyosa, J. a. Vinagre, and A. M. Jorge. Incremental matrix co-factorization for recommender systems with implicit feedback. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, pages 1413–1418. International World Wide Web Conferences Steering Committee, 2018.

[13] J.-Y. Audibert and S. Bubeck. Regret bounds and minimax policies under partial monitoring. *J. Mach. Learn. Res.*, 11:2785–2836, Dec. 2010.

[14] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2-3):235–256, May 2002.

[15] F. Bakalov, M.-J. Meurs, B. König-Ries, B. Sateli, R. Witte, G. Butler, and A. Tsang. An approach to controlling user models and personalization effects in recommender systems. In *Proc. of the 2013 International Conference on Intelligent User Interfaces*, pages 49–56, 2013.

[16] A. Bar, L. Rokach, G. Shani, B. Shapira, and A. Schclar. Improving simple collaborative filtering models using ensemble methods. In Z.-H. Zhou, F. Roli, and J. Kittler, editors, *Multiple Classifier Systems: 11th International Workshop, MCS 2013, Nanjing, China, May 15-17, 2013. Proceedings*, pages 1–12. Springer Berlin Heidelberg, 2013.

[17] V. A. Barger and L. Labrecque. An integrated marketing communications perspective on social media metrics. *International Journal of Integrated Marketing Communications*, May 2013.

[18] S. Barocas and H. Nissenbaum. Big data end run around anonymity and consent. In J. Lane, V. Stodden, S. Bender, and H. Nissenbaum, editors, *Privacy, Big Data, and the Public Good. Frameworks for Engagement*, chapter 2, pages 44–75. Cambridge University Press, 2014.

[19] J. Beel and S. Langer. A comparison of offline evaluations, online evaluations, and user studies in the context of research-paper recommender systems. In S. Kapidakis, C. Mazurek, and M. Werla, editors, *Research and Advanced Technology for Digital Libraries*, pages 153–168. Springer International Publishing, 2015.

[20] J. Beel, S. Langer, M. Genzmehr, B. Gipp, C. Breitinger, and A. Nürnberger. Research paper recommender system evaluation: A quantitative literature survey. In *Proc. of the International Workshop on Reproducibility and Replication in Recommender Systems Evaluation*, RepSys '13, 2013.

[21] D. Ben-Shimon, A. Tsikinovsky, M. Friedmann, B. Shapira, L. Rokach, and J. Hoerle. Recsys challenge 2015 and the yoochoose dataset. In *Proc. of the 9th ACM Conference on Recommender Systems*, RecSys '15, pages 357–358. ACM, 2015.

[22] S. Berkovsky, N. Borisov, Y. Eytani, T. Kuflik, and F. Ricci. Examining users' attitude towards privacy preserving collaborative filtering. In *Proc. of DM. UM*, pages 16–22, 2007.

[23] L. Bernardi, J. Kamps, J. Kiseleva, and M. J. I. Müller. The continuous cold start problem in e-commerce recommender systems. *CoRR*, abs/1508.01177, June 2015.

[24] A. J. Biega, K. P. Gummadi, and G. Weikum. Equity of attention: Amortizing individual fairness in rankings. In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '18, pages 405–414. ACM, 2018.

[25] A. Bifet, G. de Francisci Morales, J. Read, G. Holmes, and B. Pfahringer. Efficient online evaluation of big data stream classifiers. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 59–68. ACM, 2015.

[26] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. Moa: Massive online analysis. *J. Mach. Learn. Res.*, 11:1601–1604, Aug. 2010.

[27] D. Bouneffouf and R. Féraud. Multi-armed bandit problem with known trend. *Neurocomput.*, 205(C):16–21, Sept. 2016.

[28] D. Bouneffouf, R. Laroche, T. Urvoy, R. Feraud, and R. Allesiardo. Contextual bandit for active learning: Active thompson sampling. In C. K. Loo, K. S. Yap, K. W. Wong, A. Teoh, and K. Huang, editors, *Neural Information Processing: 21st International Conference, ICONIP 2014, Kuching, Malaysia, November 3-6, 2014. Proceedings, Part I*, pages 405–412. Springer International Publishing, 2014.

[29] B. Brodén, M. Hammar, B. J. Nilsson, and D. Paraschakis. Bandit algorithms for e-commerce recommender systems: Extended abstract. In *Proc. of the 11th ACM Conference on Recommender Systems*, RecSys '17, pages 349–349. ACM, 2017.

[30] B. Brodén, M. Hammar, B. J. Nilsson, and D. Paraschakis. An ensemble recommender system for e-commerce. In *26th Benelux Conference on Machine Learning (Benelearn)*, 2017.

[31] B. Brodén, M. Hammar, B. J. Nilsson, and D. Paraschakis. Ensemble recommendations via thompson sampling: An experimental study within e-commerce. In *23rd International Conference on Intelligent User Interfaces*, IUI '18, pages 19–29. ACM, 2018.

[32] B. Brodén, M. Hammar, B. J. Nilsson, and D. Paraschakis. A bandit-based ensemble framework for exploration/exploitation of diverse recommendation components: An experimental study within e-commerce. *ACM Trans. Interact. Intell. Syst.*, 10(1):4:1–4:32, Aug. 2019.

[33] B. Brost, R. Mehrotra, and T. Jehan. The music streaming sessions dataset. In *The World Wide Web Conference*, WWW '19, pages 2594–2600. ACM, 2019.

[34] R. Burke. Multisided fairness for recommendation. *CoRR*, abs/1707.00093, 2017.

[35] R. Burke, M. P. O'Mahony, and N. J. Hurley. Robust collaborative recommendation. In F. Ricci, L. Rokach, and B. Shapira, editors, *Recommender Systems Handbook*, pages 961–995. Springer US, 2015.

[36] R. Burke, N. Sonboli, and A. Ordonez-Gauger. Balanced neighborhoods for multi-sided fairness in recommendation. In S. A. Friedler and C. Wilson, editors, *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, volume 81 of *Proceedings of Machine Learning Research*, pages 202–214, 2018.

[37] J. Burrell. How the machine 'thinks': Understanding opacity in machine learning algorithms. *Big Data & Society*, 3(1), 2016.

[38] G. Burtini, J. Loeppky, and R. Lawrence. Improving online marketing experiments with drifting multi-armed bandits. In *Proc. of the 17th International Conference on Enterprise Information Systems - Volume 1*, ICEIS 2015, pages 630–636. SCITEPRESS - Science and Technology Publications, Lda, 2015.

[39] R. Cañamares, M. Redondo, and P. Castells. Multi-armed recommender system bandit ensembles. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, pages 432–436. ACM, 2019.

[40] P. Campos, F. Díez, and I. Cantador. Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. *User Modeling and User-Adapted Interaction*, 24(1):67–119, Oct. 2014.

[41] P. Cappanera and M. Trubian. A local-search-based heuristic for the demand-constrained multidimensional knapsack problem. *INFORMS J. on Computing*, 17(1):82–98, Jan. 2005.

[42] A. Caprara, H. Kellerer, U. Pferschy, and D. Pisinger. Approximation algorithms for knapsack problems with cardinality constraints. *European Journal of Operat. Research*, 123(2):333–345, 2000.

[43] S. Caron and S. Bhagat. Mixing bandits: A recipe for improved cold-start recommendations in a social network. In *Proc. of the 7th Workshop on Social Network Mining and Analysis*, SNAKDD '13, pages 11:1–11:9. ACM, 2013.

[44] S. Chang, Y. Zhang, J. Tang, D. Yin, Y. Chang, M. A. Hasegawa-Johnson, and T. S. Huang. Streaming recommender systems. In *Proceedings of the 26th International Conference on World Wide Web*, WWW âĂŹ17, page 381âĂŞ389, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee.

[45] O. Chapelle and L. Li. An empirical evaluation of thompson sampling. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2249–2257. Curran Associates, Inc., 2011.

[46] G. Chaslot and A. Gorbatai. Does youtubeâĂŹs a.i. favor trump and fake news? Online: **https://medium.com/the-graph/youtubes-ai-is-neutral-towards-clicks-but-is-biased-towards-people-and-ideas-3a2f643dea9a#.gce3bifs2**, 2016. Accessed: 2017-02-05.

[47] S. P. Chatzis, P. Christodoulou, and A. S. Andreou. Recurrent latent variable networks for session-based recommendation. In *Proceedings of the 2Nd Workshop on Deep Learning for Recommender Systems*, DLRS 2017, pages 38–45. ACM, 2017.

[48] L. Cohen, L. Manion, and K. Morrison. Research methods in education. New York: Routledge/Taylor & Francis Group, 6th edition, 2007.

[49] J. Cook. Ethics of data mining. **http://scholarworks.rit.edu/article/441**, 2005. Accessed: 2017-11-19.

[50] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *Proc. of the 2008 International Conference on Web Search and Data Mining*, WSDM '08, pages 87–94. ACM, 2008.

[51] P. Cremonesi, F. Garzotto, and R. Turrin. Investigating the persuasion potential of recommender systems from a quality perspective: An empirical study. *ACM Trans. Interact. Intell. Syst.*, 2(2):11:1–11:41, 2012.

[52] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 39–46. ACM, 2010.

[53] P. Cremonesi, R. Turrin, and F. Airoldi. Hybrid algorithms for recommending new items. In *Proc. of the 2Nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, HetRec '11, pages 33–40. ACM, 2011.

[54] M. F. Dacrema, P. Cremonesi, and D. Jannach. Are we really making much progress? a worrying analysis of recent neural recommendation approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, pages 101–109. ACM, 2019.

[55] E. Davoodi, K. Kianmehr, and M. Afsharchi. A semantic social network-based expert recommender system. *Applied Intelligence*, 39(1):1–13, 2013.

[56] D. DeCoste. Collaborative prediction using ensembles of maximum margin matrix factorizations. In *Proc. of the 23rd International Conference on Machine Learning*, ICML '06, pages 249–256. ACM, 2006.

[57] M. Denscombe. The good research guide: for small-scale social research projects (4th edition). Open University Press, 2010.

[58] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, Jan. 2004.

[59] C. Desrosiers and G. Karypis. *A Comprehensive Survey of Neighborhood-based Recommendation Methods*, pages 107–144. Springer US, 2011.

[60] T. Di Noia and V. C. Ostuni. Recommender systems and linked open data. In W. Faber and A. Paschke, editors, *Reasoning Web. Web Logic Rules: 11th International Summer School 2015, Berlin, Germany, July 31- August 4, 2015, Tutorial Lectures.*, pages 88–113. Springer International Publishing, 2015.

[61] N. Diakopoulos. Algorithmic accountability reporting: On the investigation of black boxes. *Tow Center for Digital Journalism A Tow/Knight Brief*, 2014.

[62] N. Diakopoulos, S. Friedler, M. Arenas, S. Barocas, M. Hay, B. Howe, H. Jagadish, K. Unsworth, A. Sahuguet, S. Venkatasubramanian,

C. Wilson, C. Yu, and B. Zevenbergen. Principles for accountable algorithms and a social impact statement for algorithms. Online: **https://www.fatml.org/resources/principles-for-accountable-algorithms/**. Accessed: 2017-10-25.

[63] E. Diaz-Aviles, L. Drumond, L. Schmidt-Thieme, and W. Nejdl. Real-time top-n recommendation in social streams. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, pages 59–66. ACM, 2012.

[64] T. G. Dietterich. Ensemble methods in machine learning. In *Proc. of the First International Workshop on Multiple Classifier Systems*, MCS '00, pages 1–15. Springer-Verlag, 2000.

[65] P. Domingos and G. Hulten. Mining high-speed data streams. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '00, pages 71–80. ACM, 2000.

[66] T. Dunning. Accurate methods for the statistics of surprise and coincidence. *Comput. Linguist.*, 19(1):61–74, 1993.

[67] T. Dunning. Surprise and coincidence - musings from the long tail. Online: **http://tdunning.blogspot.se/2008/03/surprise-and-coincidence.html**, mar 2008. Accessed: 2015-03-09.

[68] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel. Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 214–226. ACM, 2012.

[69] C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

[70] E. A. Dyl. A note on price discrimination by academic journals. *The Library Quarterly: Information, Community, Policy*, 53(2):317–346, 1983.

[71] S. Ebneyamini and M. R. S. Moghadam. Toward developing a framework for conducting case study research. *International Journal of Qualitative Methods*, 17(1), 2018.

[72] M. D. Ekstrand, R. Burke, and F. Diaz. Fairness and discrimination in retrieval and recommendation. In *Proceedings of the 42Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, pages 1403–1404. ACM, 2019.

[73] Y. El Madani El Alami, E. H. Nfaoui, and B. O. E. Toward an effective hybrid collaborative filtering: A new approach based on matrix factorization and heuristic-based neighborhood. In *Intelligent Systems and Computer Vision (ISCV)*, pages 1–8, 2015.

[74] . Executive Office of the President. Big data: Seizing opportunities, preserving values. Online: **`www.whitehouse.gov/sites/default/files/docs/big_data_privacy_report_may_1_2014.pdf/`**, 2014. Accessed: 2016-04-09.

[75] C. Fagan. Algorithmic accountability. Technical report, World Wide Web Foundation, July 2017.

[76] K. Faisal, T. Calders, and M. Pechenizkiy. Techniques for discrimination-free predictive models. In B. Custers, T. Calders, B. Schermer, and T. Zarsky, editors, *Discrimination and Privacy in the Information Society*, chapter 12, pages 223–239. Springer Berlin Heidelberg, 2013.

[77] C. Z. Felício, K. V. Paixão, C. A. Barcelos, and P. Preux. A multi-armed bandit model selection for cold-start user recommendation. In *Proc. of the 25th Conference on User Modeling, Adaptation and Personalization*, UMAP '17, pages 32–40. ACM, 2017.

[78] E. Felten. On the ethics of a/b testing. Online: **`freedom-to-tinker.com/blog/felten/on-the-ethics-of-ab-testing/`**, 2014. Accessed: 2016-05-15.

[79] R. Fisman, S. S. Iyengar, E. Kamenica, and I. Simonson. Gender differences in mate selection: Evidence from a speed dating experiment. *Quarterly Journal of Economics*, 121(2):673–697, 2006.

[80] R. Fisman, S. S. Iyengar, E. Kamenica, and I. Simonson. Racial preferences in dating. *The Review of Economic Studies*, 75(1):117–132, 2008.

[81] L. Floridi and J. W. Sanders. On the morality of artificial agents. *Minds Mach.*, 14(3):349–379, Aug. 2004.

[82] A. Friedman, B. Knijnenburg, K. Vanhecke, L. Martens, and S. Berkovsky. Privacy aspects of recommender systems. In F. Ricci, L. Rokach, and B. Shapira, editors, *Recommender Systems Handbook (2nd edition)*, chapter 19, pages 649–688. Springer Science+Business Media New York, 2015.

[83] E. Frigó, R. Pálovics, D. Kelen, L. Kocsis, and A. Benczúr. Alpenglow: Open source recommender framework with time-aware learning and evaluation. In *2017 Poster Track of the 11th ACM Conference on Recommender Systems, Poster-Recsys 2017*, pages 1–2. CEUR-WS.org, 2017.

[84] P. Gajane and M. Pechenizkiy. On formalizing fairness in prediction with machine learning. *CoRR*, abs/1710.03184, 2017.

[85] J. Gama, R. SebastiÃčo, and P. P. Rodrigues. On evaluating stream learning algorithms. *Machine Learning*, 90(3):161–168, 2013.

[86] Z. Gantner, S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Mymedialite: A free recommender system library. In *Proc. of the 5th ACM Conference on Recommender Systems*, RecSys 2011, 2011.

[87] A. Garivier and O. Cappé. The KL-UCB algorithm for bounded stochastic bandits and beyond. In *The 24th Annual Conference on Learning Theory*, pages 359–376, 2011.

[88] A. Garivier and E. Moulines. On upper-confidence bound policies for switching bandit problems. In *Proc. of the 22Nd International Conference on Algorithmic Learning Theory*, ALT'11, pages 174–188. Springer-Verlag, 2011.

[89] B. Gert. Common morality: Deciding what to do. Oxford University Press, 2004.

[90] S. Geuens. Factorization machines for hybrid recommendation systems based on behavioral, product, and customer data. In *Proc. of the 9th ACM Conference on Recommender Systems*, RecSys '15, pages 379–382. ACM, 2015.

[91] A. Gopalan, S. Mannor, and Y. Mansour. Thompson sampling for complex online problems. In *Proc. of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages I–100–I–108. JMLR.org, 2014.

[92] S. Gosepath. Equality. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, spring 2011 edition, 2011.

[93] J. Grimmelmann. Illegal, immoral, and mood-altering. how facebook and okcupid broke the law when they experimented on users. Online: **medium.com/@JamesGrimmelmann/illegal-unethical-and-mood-altering-8b93af772688**, 2014. Accessed: 2016-05-14.

[94] F. Guillou. *On recommendation systems in a sequential context*. PhD thesis, Université Charles de Gaulle - Lille III, Dec. 2016.

[95] A. Gunawardana and G. Shani. Evaluating recommender systems. In F. Ricci, L. Rokach, and B. Shapira, editors, *Recommender Systems Handbook*, pages 265–308. Springer US, 2015.

[96] L. Guo, H. Yin, Q. Wang, T. Chen, A. Zhou, and N. Quoc Viet Hung. Streaming session-based recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pages 1569–1577. ACM, 2019.

[97] J. Hagen-Zanker and R. Mallett. How to do a rigorous, evidence-focused literature review in international development. Technical report, Overseas Development Institute, 2013.

[98] J. Han, J. Pei, Y. Yin, and R. Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, 8(1):53–87, Jan 2004.

[99] A. Hannak, G. Soeller, D. Lazer, A. Mislove, and C. Wilson. Measuring price discrimination and steering on e-commerce web sites. In *Proc. of the 2014 Conference on Internet Measurement Conference*, IMC '14, pages 305–318, 2014.

[100] M. Hardt, E. Price, E. Price, and N. Srebro. Equality of opportunity in supervised learning. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3315–3323. Curran Associates, Inc., 2016.

[101] N. Hariri, B. Mobasher, and R. Burke. Adapting to user preference changes in interactive recommendation. In *Proc. of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 4268–4274. AAAI Press, 2015.

[102] C. Hartland, S. Gelly, N. Baskiotis, O. Teytaud, and M. Sebag. Multi-armed bandit, dynamic environments and meta-bandits. In *Online Trading of Exploration and Exploitation, NIPS 2006 Workshop*, 2006.

[103] B. Heitmann and C. Hayes. Using linked data to build open, collaborative recommender systems. In *AAAI Spring Symposium: Linked Data Meets Artificial Intelligence*, pages 76–81, 2010.

[104] P. J. Henrichsen. Ethical intelligence in social recommender systems. In *Proceeding of the 15th International Conference on Intelligent User Interfaces*, 2010.

[105] J. L. Herlocker, J. A. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. In *Proc. of the 2000 ACM Conference on Computer Supported Cooperative Work*, CSCW '00, pages 241–250. ACM, 2000.

[106] A. Hevner and S. Chatterjee. Design science research in information systems. In *Design Research in Information Systems: Theory and Practice*, pages 9–22. Springer US, 2010.

[107] A. R. Hevner, S. T. March, J. Park, and S. Ram. Design science in information systems research. *MIS Q.*, 28(1):75–105, 2004.

[108] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk. Session-based recommendations with recurrent neural networks. In *CoRR*, abs/1511.06939, 2015.

[109] B. Hidasi, M. Quadrana, A. Karatzoglou, and D. Tikk. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 241–248. ACM, 2016.

[110] B. Hidasi and D. Tikk. Context-aware item-to-item recommendation within the factorization framework. In *Proceedings of the 3rd Workshop on Context-awareness in Retrieval and Recommendation*, CaRR '13, pages 19–25. ACM, 2013.

[111] D. Howard and I. Muntean. A minimalist model of the artificial autonomous moral agent (aama). In *SSS-16 Symposium Technical Reports. Association for the Advancement of Artificial Intelligence*. AAAI, 2016.

[112] C.-C. Hsieh, J. Neufeld, T. King, and J. Cho. Efficient approximate thompson sampling for search query recommendation. In *Proc. of the 30th Annual ACM Symposium on Applied Computing*, SAC '15, pages 740–746. ACM, 2015.

[113] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Proc. of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pages 263–272, 2008.

[114] K.-H. Huang and H.-T. Lin. Linear upper confidence bound algorithm forźcontextual bandit problem with piled rewards. In *Proceedings, Part II, of the 20th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining - Volume 9652*, PAKDD 2016, pages 143–155. Springer-Verlag New York, Inc., 2016.

[115] Z. Huang, D. Zeng, and H. Chen. A comparative study of recommendation algorithms for e-commerce applications. *IEEE Intelligent Systems*, 2006.

[116] J. S. Hunter and T. H. Naylor. Experimental designs for computer simulation experiments. *Management Science*, 16(7):422–434, 1970.

[117] J. A. Hutson, J. G. Taft, S. Barocas, and K. Levy. Debiasing desire: Addressing bias & discrimination on intimate platforms. *Proc. ACM Hum.-Comput. Interact.*, 2(CSCW):73:1–73:18, Nov. 2018.

[118] M. Jahrer, A. Töscher, and R. Legenstein. Combining predictions for accurate recommender systems. In *Proc. of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 693–702. ACM, 2010.

[119] D. Jannach. Keynote: Session-based recommendation – challenges and recent advances. In F. Trollmann and A.-Y. Turhan, editors, *KI 2018: Advances in Artificial Intelligence*, pages 3–7. Springer International Publishing, 2018.

[120] D. Jannach and M. Ludewig. When recurrent neural networks meet the neighborhood for session-based recommendation. In *Proc. of the Eleventh ACM Conference on Recommender Systems*, RecSys '17, pages 306–310. ACM, 2017.

[121] D. Jannach, M. Ludewig, and L. Lerche. Session-based item recommendation in e-commerce: on short-term intents, reminders, trends and discounts. *User Modeling and User-Adapted Interaction*, 27(3):351–392, Dec 2017.

[122] D. Jannach, P. Resnick, A. Tuzhilin, and M. Zanker. Recommender systems — beyond matrix completion. *Commun. ACM*, 59(11):94–102, Oct. 2016.

[123] S. Joel, P. W. Eastwick, and E. J. Finkel. Is romantic desire predictable? machine learning applied to initial romantic attraction. *Psychological Science*, 28(10):1478–1489, 2017.

[124] C. C. Johnson. Logistic matrix factorization for implicit feedback data. In *NIPS Workshop on Distributed Matrix Computations*, 2014.

[125] M. Joseph, M. Kearns, J. Morgenstern, S. Neel, and A. Roth. Meritocratic fairness for infinite and contextual bandits. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '18, pages 158–163. ACM, 2018.

[126] M. Jugovac, D. Jannach, and M. Karimi. Streamingrec: A framework for benchmarking stream-based news recommenders. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, pages 269–273. ACM, 2018.

[127] S. Kabbur, X. Ning, and G. Karypis. Fism: factored item similarity models for top-n recommender systems. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 659–667. ACM, 2013.

[128] I. Kamehkhosh, D. Jannach, and M. Ludewig. A comparison of frequent pattern techniques and a deep learning method for session-based recommendation. In *TempRec Workshop at ACM RecSys*, 2017.

[129] T. Kamishima and S. Akaho. Personalized pricing recommender system: Multi-stage epsilon-greedy approach. In *Proc. of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, pages 57–64. ACM, 2011.

[130] T. Kamishima, S. Akaho, and H. Asoh. Enhancement of the neutrality in recommendation. In *In Proc. of the 2nd Workshop on Human Decision Making in Recommender Systems*, pages 8–14, 2012.

[131] V. Kanade, B. Mcmahan, and B. Brent. Sleeping experts and bandits with stochastic action availability and adversarial rewards. In *In Proc. of the Twelfth International Conference on Artificial Intelligence and Statistics, number 5*, pages 272–279, 2009.

[132] C. Karako and P. Manggala. Using image fairness representations in diversity-based re-ranking for recommendations. In *Adjunct Publication of the 26th Conference on User Modeling, Adaptation and Personalization*, UMAP '18, pages 23–28. ACM, 2018.

[133] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver. Multiverse recommendation: N-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys âĂŹ10, page 79âĂŞ86. ACM, 2010.

[134] A. Karatzoglou, L. Baltrunas, and Y. Shi. Learning to rank for recommender systems. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 493–494. ACM, 2013.

[135] E. Kaufmann, N. Korda, and R. Munos. Thompson sampling: An asymptotically optimal finite-time analysis. In *Proc. of the 23rd International Conference on Algorithmic Learning Theory*, ALT'12, pages 199–213. Springer-Verlag, 2012.

[136] J. Kawale, H. H. Bui, B. Kveton, L. Tran-Thanh, and S. Chawla. Efficient thompson sampling for online ï£ijmatrix factorization recommendation.

In *Advances in Neural Information Processing Systems 28*, pages 1297–1305. Curran Associates, Inc., 2015.

[137] B. Kille, F. Hopfgartner, T. Brodt, and T. Heintz. The plista dataset. In *Proceedings of the 2013 International News Recommender Systems Workshop and Challenge*, NRS '13, pages 16–23. ACM, 2013.

[138] B. Kille, A. Lommatzsch, R. Turrin, A. Serény, M. Larson, T. Brodt, J. Seiler, and F. Hopfgartner. Stream-based recommendations: Online and offline evaluation as a service. In J. Mothe, J. Savoy, J. Kamps, K. Pinel-Sauvagnat, G. Jones, E. San Juan, L. Capellato, and N. Ferro, editors, *Experimental IR Meets Multilinguality, Multimodality, and Interaction*, pages 497–517. Springer International Publishing, 2015.

[139] C. Kim and J. Kim. A recommendation algorithm using multi-level association rules. In *Proc. of the 2003 IEEE/WIC International Conference on Web Intelligence*, WI '03, 2003.

[140] T. Kitazawa. Flurs: A python library for online item recommendation. **https://takuti.me/note/flurs/**, 2017. Accessed: 2019-11-03.

[141] J. Kleinberg, S. Mullainathan, and M. Raghavan. Inherent trade-offs in the fair determination of risk scores, 2016.

[142] R. Kleinberg, A. Niculescu-Mizil, and Y. Sharma. Regret bounds for sleeping experts and bandits. *Mach. Learn.*, 80(2-3):245–272, Sept. 2010.

[143] P. Knees, Y. Deldjoo, F. Bakhshandegan Moghaddam, J. Adamczak, G.-P. Leyson, and P. Monreal. Recsys challenge 2019: Session-based hotel recommendations. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19. ACM, 2019.

[144] B. Knijnenburg. Simplifying privacy decisions: Towards interactive and adaptive solutions. In *Decisions@ RecSys*, pages 40–41, 2013.

[145] B. Knijnenburg. *A User-Tailored Approach to Privacy Decision Support*. PhD thesis, University of California, 2015.

[146] T. Kocák, M. Valko, R. Munos, and S. Agrawal. Spectral thompson sampling. In *Proc. of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, pages 1911–1917. AAAI Press, 2014.

[147] A. Koene, E. Perez, C. J. Carter, R. Statache, S. Adolphs, C. O'Malley, T. Rodden, and D. McAuley. Ethics of personalized information filtering. In *Second International Conference on Internet Science, INSCI*, pages 123–132, 2015.

[148] A. E. Kohlhase and M. Kohlhase. Semantic transparency in user assistance systems. In *Proc. of the 27th ACM International Conference on Design of Communication*, SIGDOC '09, pages 89–96. ACM, 2009.

[149] P. Kohli, M. Salek, and G. Stoddard. A fast bandit algorithm for recommendations to users with heterogeneous tastes. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, AAAIâĂŹ13, page 1135âĂŞ1141. AAAI Press, 2013.

[150] J. Komiyama, J. Honda, and H. Nakagawa. Optimal regret analysis of thompson sampling in stochastic multi-armed bandit problem with multiple plays. In *Proc. of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 1152–1161. JMLR.org, 2015.

[151] Y. Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proc. of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 426–434. ACM, 2008.

[152] Y. Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Trans. Knowl. Discov. Data*, 4(1):1:1–1:24, Jan. 2010.

[153] Y. Koren and R. Bell. *Advances in Collaborative Filtering*, pages 145–186. Springer US, 2011.

[154] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, Aug 2009.

[155] M. J. Kusner, J. Loftus, C. Russell, and R. Silva. Counterfactual fairness. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4066–4076. Curran Associates, Inc., 2017.

[156] B. Kveton, C. Szepesvári, Z. Wen, and A. Ashkan. Cascading bandits: Learning to rank in the cascade model. In *Proc. of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 767–776. JMLR.org, 2015.

[157] M. Kyriakidi, K. Stefanidis, and Y. Ioannidis. On achieving diversity in recommender systems. In *Proceedings of the ExploreDB'17*, ExploreDB'17, pages 4:1–4:6. ACM, 2017.

[158] A. Lacerda. Multi-objective ranked bandits for recommender systems. *Neurocomput.*, 246(C):12–24, July 2017.

[159] T. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Adv. Appl. Math.*, 6(1):4–22, Mar. 1985.

[160] N. Lathia. *Evaluating collaborative filtering over time*. PhD thesis, UCL (University College London), 2010.

[161] D. Li, C. Chen, Q. Lv, L. Shang, Y. Zhao, T. Lu, and N. Gu. An algorithm for efficient privacy-preserving item-based collaborative filtering. *Future Generation Computer Systems*, 55:311 – 320, 2016.

[162] G. Li and Q. Chen. Exploiting explicit and implicit feedback for personalized ranking. *Mathematical Problems in Engineering*, pages 1 – 11, 2016.

[163] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 1419–1428. ACM, 2017.

[164] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proc. of the 19th International Conference on World Wide Web*, WWW '10, pages 661–670. ACM, 2010.

[165] L. Li, W. Chu, J. Langford, and X. Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proc. of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 297–306. ACM, 2011.

[166] S. Li, A. Karatzoglou, and C. Gentile. Collaborative filtering bandits. In *Proc. of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 539–548. ACM, 2016.

[167] S. Li, B. Wang, S. Zhang, and W. Chen. Contextual combinatorial cascading bandits. In *Proc. of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1245–1253. JMLR.org, 2016.

[168] W. Liu and R. Burke. Personalizing fairness-aware re-ranking. *CoRR*, abs/1809.02921, 2018.

[169] Y. Liu, G. Radanovic, C. Dimitrakakis, D. Mandal, and D. C. Parkes. Calibrated fairness in bandits. In *Proceedings of the 4th Workshop on Fairness, Accountability, and Transparency in Machine Learning (Fat/ML 2017)*, 2017.

[170] A. Lommatzsch and S. Albayrak. Real-time recommendations for user-item streams. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, SAC '15, pages 1039–1046. ACM, 2015.

[171] P. Lops, M. de Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 73–105. Springer US, 2011.

[172] J. Louëdec, M. Chevalier, J. Mothe, A. Garivier, and S. Gerchinovitz. A multiple-play bandit algorithm applied to recommender systems. In *Proc. of the Twenty-Eighth International Florida Artificial Intelligence Research Society Conference, FLAIRS*, pages 67–72, 2015.

[173] M. Ludewig and D. Jannach. Evaluation of session-based recommendation algorithms. *User Modeling and User-Adapted Interaction*, 28(4-5):331–390, Oct 2018.

[174] M. Ludewig, N. Mauro, S. Latifi, and D. Jannach. Empirical analysis of session-based recommendation algorithms, 2019.

[175] X. Luo, Y. Xia, and Q. Zhu. Incremental collaborative filtering recommender based on regularized matrix factorization. *Knowledge-Based Systems*, 27:271 – 280, 2012.

[176] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data*, 1(1), Mar. 2007.

[177] C. D. Manning, P. Raghavan, and H. Schutze. Evaluation in information retrieval. In *Introduction to Information Retrieval*, pages 151–175. Cambridge University Press, 2008.

[178] S. T. March and G. F. Smith. Design and natural science research on information technology. *Decis. Support Syst.*, 15(4):251–266, 1995.

[179] M. Marshall. Aggregate knowledge raises $5m from kleiner, on a roll. **http://venturebeat.com/2006/12/10/aggregate-knowledge-raises-5m-from-kleiner-on-a-roll/**, Dec. 2006. Accessed: 2015-03-12.

[180] S. Martello and P. Toth. Algorithms for knapsack problems. *North-Holland Mathematics Studies*, 132:213–257, 1987.

[181] J. Mary, R. Gaudel, and P. Preux. Bandits warm-up cold recommender systems. *CoRR*, abs/1407.2806, June 2014.

[182] S. Masoudnia and R. Ebrahimpour. Mixture of experts: A literature survey. *The Artificial Intelligence Review*, 42(2):275–293, 2014.

[183] M. Massoud and M. Abo-Rizka. A conceptual model of personalized pricing recommender system based on customer online behavior. *International Journal of Computer Science & Network Security*, 12(6):129–133, 2012.

[184] P. Matuszyk. *Selective Learning for Recommender Systems*. PhD thesis, Otto-von-Guericke-UniversitÃďt Magdeburg, 2017.

[185] A. M. McDonald and C. L. Faith. The cost of reading privacy policies. *I/S: A Journal of Law and Policy for the Information Society*, 4:540–565, 2008.

[186] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan. A survey on bias and fairness in machine learning, 2019.

[187] J. Mellor and J. Shapiro. Thompson sampling in switching environments with bayesian online change point detection. In *Proc. of the Sixteenth International Conference on Artificial Intelligence and Statistics*, pages 442–450, 2013.

[188] J. Metcalf. Letter on proposed changes to the common rule. Online: **bdes.datasociety.net/council-output/letter-on-proposed-changes-to-the-common-rule/**, December 2015. Accessed: 2016-04-08.

[189] F. Meyer. *Recommender systems in industrial contexts*. PhD thesis, Université Grenoble, Jan. 2012.

[190] S. Milano, M. Taddeo, and L. Floridi. Ethical aspects of multi-stakeholder recommendation systems. Available at SSRN: **https://ssrn.com/abstract=3493202**, nov 2019. Accessed: 2019-09-12.

[191] S. Milano, M. Taddeo, and L. Floridi. Recommender systems and their ethical challenges. Available at SSRN: **https://ssrn.com/abstract=3378581**, apr 2019. Accessed: 2019-09-12.

[192] A. Mills, G. Durepos, and E. Wiebe. Comparative case study. *Encyclopedia of Case Study Research*, 1-0, 2010.

[193] T. Mine, T. Kakuta, and A. Ono. Reciprocal recommendation for job matching with bidirectional feedback. In *2013 Second IIAI International Conference on Advanced Applied Informatics*, pages 39–44. IEEE, 2013.

[194] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa. Effective personalization based on association rule discovery from web usage data. In *Proc. of the 3rd international workshop on Web information and data management*, pages 9–15. ACM, 2001.

[195] B. Mobasher, Honghua Dai, Tao Luo, and M. Nakagawa. Using sequential and non-sequential patterns in predictive web usage mining tasks. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 669–672, 2002.

[196] S. Mohseni, E. Ragan, and X. Hu. Open issues in combating fake news: Interpretability as an opportunity, 2019.

[197] R. Montaño, A. Kutzke, J. Wojciechowski, R. Fontana, J. Marynowski, and A. Schmitz. Which products would you like to see? user preferences for top-n e-commerce products recommendation in different shopping moments. *International Journal of Computer and Information Technology*, 7:190, June 2018.

[198] J. Montiel, J. Read, A. Bifet, and T. Abdessalem. Scikit-multiflow: A multi-output streaming framework. *Journal of Machine Learning Research*, 19(72):1–5, 2018.

[199] J. H. Moor. Just consequentialism and computing. *Ethics and Inf. Technol.*, 1(1):61–65, Jan. 1998.

[200] J. H. Moor. The nature, importance, and difficulty of machine ethics. *IEEE Intelligent Systems*, 21(4):18–21, July 2006.

[201] C. Musto, G. Semeraro, M. de Gemmis, and P. Lops. Learning word embeddings from wikipedia for content-based recommender systems. In N. Ferro, F. Crestani, M.-F. Moens, J. Mothe, F. Silvestri, G. M. Di Nunzio, C. Hauff, and G. Silvello, editors, *Advances in Information Retrieval: 38th European Conference on IR Research, ECIR 2016, Padua, Italy, March 20–23, 2016. Proceedings*, pages 729–734. Springer International Publishing, 2016.

[202] M. Nakagawa and B. Mobasher. Impact of site characteristics on recommendation models based on association rules and sequential patterns. In *Proceedings of the IJCAI*, volume 3, pages 1–10, 2003.

[203] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *Proc. of the 2008 IEEE Symposium on Security and Privacy*, pages 111–125. IEEE Computer Society, 2008.

[204] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *30th IEEE Symposium on Security and Privacy*, pages 173–187. IEEE, 2009.

[205] O. Nasraoui, J. Cerwinske, C. Rojas, and F. Gonzalez. Performance of recommendation systems in dynamic streaming environments. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, 2007.

[206] N. Natarajan, D. Shin, and I. S. Dhillon. Which app will you use next?: Collaborative filtering with interactional context. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 201–208. ACM, 2013.

[207] M.-Q. Nguyen. Multi-armed bandit problem and its applications in intelligent tutoring systems. Master's thesis, ÃĽcole Polytechnique, 2014.

[208] H. Nissenbaum. Computing and accountability. *Commun. ACM*, 37(1):72–80, Jan. 1994.

[209] H. Nissenbaum. Human values and the design of computer technology. chapter Accountability in a Computerized Society, pages 41–64. Center for the Study of Language and Information, 1997.

[210] K. Notopoulos. The dating app that knows you secretly aren't into guys from other races. **`https://www.buzzfeednews.com/article/katienotopoulos/coffee-meets-bagel-racial-preferences`**, Jan. 2016. Accessed: 2019-06-26.

[211] B. J. Oates. Researching information systems and computing. Sage Publications Ltd., 2006.

[212] J. Obar. Big data and the phantom public: Walter lippmann and the fallacy of data privacy self-management. *Big Data & Society*, 2015.

[213] P. Ohm. Broken promises of privacy: Responding to the surprising failure of anonymization. *UCLA Law Review*, 57:1701, 2009.

[214] S. Owen, R. Anil, T. Dunning, and E. Friedman. Mahout in action. Manning Publications Co., 2011.

[215] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *Proc. of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pages 502–511, 2008.

[216] D. Paraschakis. Recommender systems from an industrial and ethical perspective. In *Proc. of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 463–466. ACM, 2016.

[217] D. Paraschakis. Towards an ethical recommendation framework. In *Proc. of the 11th International Conference on Research Challenges in Information Science (RCIS)*, pages 211 – 220. IEEE, 2017.

[218] D. Paraschakis and B. J. Nilsson. Flowrec: Prototyping session-based recommendations in streaming mode. In *Proc. of the 24th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, PAKDD '20. Springer, 2020.

[219] D. Paraschakis and B. J. Nilsson. Matchmaking under fairness constraints: a speed dating case study. In *Proc. of the International Workshop on Algorithmic Bias in Search and Recommendation*, Bias 2020. Springer, 2020.

[220] D. Paraschakis and B. J. Nilsson. On preferential fairness of matchmaking: a speed dating case study. In *Proc. of the 18th International Conference on the Ethical and Social Impacts of ICT*, ETHICOMP '20, 2020.

[221] D. Paraschakis, B. J. Nilsson, and J. Holländer. Comparative evaluation of top-N recommenders in e-commerce: An industrial perspective. In *Proc. of the 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 1024 – 1031. IEEE, 2015.

[222] S.-T. Park and W. Chu. Pairwise preference regression for cold-start recommendation. In *Proc. of the Third ACM Conference on Recommender Systems*, RecSys '09, pages 21–28. ACM, 2009.

[223] F. Pasquale. The black box society. the secret algorithms that control money and information. Harvard University Press, 2015.

[224] M. J. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artif. Intell. Rev.*, 13(5-6):393–408, Dec. 1999.

[225] M. J. Pazzani and D. Billsus. Content-based recommendation systems. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The Adaptive Web: Methods and Strategies of Web Personalization*, pages 325–341. Springer Berlin Heidelberg, 2007.

[226] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12:2825–2830, Nov. 2011.

[227] Pew Research Center. Racial and ethnic composition by religious group. **https://www.pewforum.org/religious-landscape-study/racial-and-ethnic-composition/**, 2014. Accessed: 2019-07-09.

[228] A. Pinsonneault and K. L. Kraemer. Survey research methodology in management information systems: An assessment. *J. Manage. Inf. Syst.*, 10(2):75–105, Sept. 1993.

[229] H. Polat and W. Du. Privacy-preserving collaborative filtering using randomized perturbation techniques. In *Proc. of the Third IEEE International Conference on Data Mining*, pages 625–. IEEE Computer Society, 2003.

[230] B. Pradel, S. Sean, J. Delporte, S. Guérif, C. Rouveirol, N. Usunier, F. Fogelman-Soulié, and F. Dufau-Joel. A case study in a recommender system based on purchase data. In *Proc. of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 377–385, 2011.

[231] C. Puschmann and E. Bozdag. Staking out the unclear ethical terrain of online social experiments. *Internet Policy Review*, 3, nov 2014.

[232] M. Quadrana. *Algorithms for Sequence-Aware Recommender Systems*. PhD thesis, Politecnico di Milano, 2017.

[233] M. Quadrana, P. Cremonesi, and D. Jannach. Sequence-aware recommender systems. *ACM Comput. Surv.*, 51(4):66:1–66:36, July 2018.

[234] M. Quadrana, A. Karatzoglou, B. Hidasi, and P. Cremonesi. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *Proc. of the Eleventh ACM Conference on Recommender Systems*, RecSys '17, pages 130–137. ACM, 2017.

[235] F. Radlinski, R. Kleinberg, and T. Joachims. Learning diverse rankings with multi-armed bandits. In *Proc. of the 25th International Conference on Machine Learning*, ICML '08, pages 784–791. ACM, 2008.

[236] N. Ramakrishnan, B. J. Keller, B. J. Mirza, A. Y. Grama, and G. Karypis. Privacy risks in recommender systems. *IEEE Internet Computing*, 5(6):54–62, 2001.

[237] A. Ramlatchan, M. Yang, Q. Liu, M. Li, J. Wang, and Y. Li. A survey of matrix completion methods for recommendation systems. *Big Data Mining and Analytics*, 1(4):308–323, December 2018.

[238] J. Read, P. Reutemann, B. Pfahringer, and G. Holmes. Meka: A multi-label/multi-target extension to weka. *J. Mach. Learn. Res.*, 17(1):667–671, Jan. 2016.

[239] S. Rendle. Factorization machines. In *Proc. of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 995–1000. IEEE Computer Society, 2010.

[240] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proc. of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, 2009.

[241] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of*

the 19th international conference on World wide web, pages 811–820. ACM, 2010.

[242] J. D. M. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proc. of the 22Nd International Conference on Machine Learning*, ICML '05, pages 713–719. ACM, 2005.

[243] J. Rishabh. When recommendation systems go bad. Online: **cds.nyu.edu/recommendation-systems-go-bad/**, 2016. Accessed: 2016-05-15.

[244] M. Rodriguez, C. Posse, and E. Zhang. Multiple objective optimization in recommender systems. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, pages 11–18. Association for Computing Machinery, 2012.

[245] M. Rodriguez and J. Watkins. Faith in the algorithm, part 2: Computational eudaemonics. In *13th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems: Part II*, pages 813–820. Springer Berlin Heidelberg, 2009.

[246] T. Rohle. Desperately seeking the consumer: Personalized search engines and the commercial exploitation of user data. Online: **journals.uic.edu/ojs/index.php/fm/article/view/2008/1883l**, sept 2007. Accessed: 2016-04-08.

[247] M. Ross. Do research ethics need updating for the digital age? *Monitor on Psychology*, 45(9):64, oct 2014.

[248] P. Runeson and M. Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(131), 2008.

[249] A. Said, A. Bellogín, and A. de Vries. A top-n recommender system evaluation protocol inspired by deployed systems. *LSRS Workshop at ACM RecSys*, 2013.

[250] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *Proc. of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, pages 1257–1264. Curran Associates Inc., 2007.

[251] J. Salter and N. Antonopoulos. Cinemascreen recommender agent: combining collaborative and content-based filtering. *IEEE Intelligent Systems*, 21(1):35–41, 2006.

[252] S. M. Sanchez. Work smarter, not harder: guidelines for designing simulation experiments. In *Proc. of the Winter Simulation Conference, 2005.*, pages 14 pp.–, Dec 2005.

[253] C. Sandvig, K. Hamilton, K. Karahalios, and C. Langbort. Auditing algorithms: Research methods for detecting discrimination on internet platforms. In *Pre-conference of the 64th annual meeting of the International Communication Association*, 2014.

[254] P. Sapiezynski, W. Zeng, R. E Robertson, A. Mislove, and C. Wilson. Quantifying the impact of user attention on fair group representation in ranked lists. In *Companion Proceedings of the 2019 World Wide Web Conference*, WWW âĂŹ19, pages 553–562. ACM, 2019.

[255] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Analysis of recommendation algorithms for e-commerce. In *Proc. of the 2Nd ACM Conference on Electronic Commerce*, EC '00, pages 158–167, 2000.

[256] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proc. of the 10th International Conference on World Wide Web*, WWW '01, pages 285–295. ACM, 2001.

[257] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Incremental singular value decomposition algorithms for highly scalable recommender systems. In *Fifth International Conference on Computer and Information Science*, pages 27–28, 2002.

[258] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Incremental singular value decomposition algorithms for highly scalable recommender systems. pages 27–28, 2002.

[259] N. A. Saxena, K. Huang, E. DeFilippis, G. Radanovic, D. C. Parkes, and Y. Liu. How do fairness definitions fare?: Examining public attitudes towards algorithmic definitions of fairness. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '19, pages 99–106. ACM, 2019.

[260] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *Proc. of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '02, pages 253–260. ACM, 2002.

[261] G. Schröder, M. Thiele, and W. Lehner. Setting goals and choosing metrics for recommender system evaluations. In *UCERSTI2 Workshop at the 5th ACM Conference on Recommender Systems*, volume 23, page 53, 2011.

[262] B. Schwartz. The paradox of choice: why more is less. ECCO New York, 1st ed. edition, 2004.

[263] S. L. Scott. A modern bayesian look at the multi-armed bandit. *Applied Stochastic Models in Business and Industry*, 26(6):639–658, 2010.

[264] M. Scriminaci, A. Lommatzsch, B. Kille, F. Hopfgartner, M. Larson, D. Malagoli, A. SerÃĺny, and T. Plumbaum. Idomaar: A framework for multi-dimensional benchmarking of recommender algorithms. In I. Guy and A. Sharma, editors, *RecSys Posters*, CEUR Workshop Proceedings. CEUR-WS.org, 2016.

[265] A. Singh and T. Joachims. Fairness of exposure in rankings. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, pages 2219–2228. ACM, 2018.

[266] S. Singh, T. Jaakkola, M. L. Littman, and C. Szepesvári. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning*, 38(3):287–308, Mar 2000.

[267] A. Slivkins. Contextual bandits with similarity information. *J. Mach. Learn. Res.*, 15(1):2533–2568, Jan. 2014.

[268] A. Slivkins, F. Radlinski, and S. Gollapudi. Ranked bandits in metric spaces: Learning diverse rankings over large document collections. *J. Mach. Learn. Res.*, 14(1):399–436, Feb. 2013.

[269] Q. Song, X. Huang, H. Ge, J. Caverlee, and X. Hu. Multi-aspect streaming tensor completion. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 435–443. ACM, 2017.

[270] K. Souali, A. El Afia, and R. Faizi. An automatic ethical-based recommender system for e-commerce. In *International Conference on Multimedia Computing and Systems*, pages 1–4. IEEE, 2011.

[271] T. Speicher, H. Heidari, N. Grgic-Hlaca, K. P. Gummadi, A. Singla, A. Weller, and M. B. Zafar. A unified approach to quantifying algorithmic unfairness: Measuring individual & group unfairness via inequality indices. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, pages 2239–2248. ACM, 2018.

[272] P. Srimani and M. M. Patil. Performance analysis of hoeffding trees in data streams by using massive online analysis framework. *International Journal of Data Mining, Modelling and Management*, 7(4):293–313, 2015.

[273] H. Steck. Calibrated recommendations. In *Proc. of the 12th ACM Conference on Recommender Systems*, pages 154–162. ACM, 2018.

[274] M. Streeter and D. Golovin. An online algorithm for maximizing submodular functions. In *Proc. of the 21st International Conference on Neural Information Processing Systems*, NIPS'08, pages 1577–1584. Curran Associates Inc., 2008.

[275] K. Subbian, C. Aggarwal, and K. Hegde. Recommendations for streaming data. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 2185–2190. ACM, 2016.

[276] L. Sweeney. Uniqueness of simple demographics in the u.s. population. Technical report, Carnegie Mellon University, School of Computer Science. Data Privacy Laboratory, Technical Report LIDAP-WP4, 2000.

[277] L. Sweeney. K-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.

[278] G. Takács, I. Pilászy, B. Németh, and D. Tikk. Major components of the gravity recommendation system. *SIGKDD Explor. Newsl.*, 9(2):80–83, Dec. 2007.

[279] G. Takács, I. Pilászy, B. Németh, and D. Tikk. Scalable collaborative filtering approaches for large recommender systems. *J. Mach. Learn. Res.*, 10:623–656, June 2009.

[280] Y. K. Tan, X. Xu, and Y. Liu. Improved recurrent neural networks for session-based recommendations. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, DLRS 2016, pages 17–22. ACM, 2016.

[281] L. Tang, Y. Jiang, L. Li, and T. Li. Ensemble contextual bandits for personalized recommendation. In *Proc. of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 73–80. ACM, 2014.

[282] T. Tang and P. Winoto. I should not recommend it to you even if you will like it: the ethics of recommender systems. *New Review of Hypermedia and Multimedia*, 22(1-2):111–138, 2016.

[283] H. Tavani. Ethics and technology: Controversies, questions and strategies for ethical computing. Wiley, 2015.

[284] W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3–4):285–294, 1933.

[285] N. Tintarev and J. Masthoff. Designing and evaluating explanations for recommender systems. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 479–510. Springer US, 2011.

[286] T. X. Tuan and T. M. Phuong. 3d convolutional networks for session-based recommendation with content features. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, RecSys '17, pages 138–146. ACM, 2017.

[287] J. Turow, L. Feldman, and K. Meltzer. Open to exploitation: America's shoppers online and offline. Technical report, Annenberg Public Policy Center of the University of Pennsylvania, Jan 2005.

[288] R. Turrin, M. Quadrana, A. Condorelli, R. Pagano, and P. Cremonesi. In *RecSys 2015 Poster Proceedings*. Dataset: **http://recsys.deib.polimi.it/datasets/**.

[289] B. Twardowski. Modelling contextual information in session-aware recommender systems with neural networks. In *Proc. of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 273–276. ACM, 2016.

[290] T. Uchiya, A. Nakamura, and M. Kudo. Algorithms for adversarial bandit problems with multiple plays. In *Proc. of the 21st International Conference on Algorithmic Learning Theory*, ALT'10, pages 375–389. Springer-Verlag, 2010.

[291] V. K. Vaishnavi and W. Kuechler, Jr. Design science research methods and patterns: Innovating information and communication technology. Auerbach Publications, 1st edition, 2007.

[292] J. Valentino-DeVries, J. Singer-Vine, and A. Soltani. Websites vary prices, deals based on users' information. Online: **www.wsj.com/articles/SB10001424127887323777204578189391813881534/**, December 2012. Accessed: 2016-05-02.

[293] S. B. Van Der Zon. Predictive performance and discrimination in unbalanced classification. Master's thesis, TU Eindhoven, 2016.

[294] S. Verma and J. Rubin. Fairness definitions explained. In *Proceedings of the International Workshop on Software Fairness*, FairWare '18, pages 1–7. ACM, 2018.

[295] S. Verwer and T. Calders. Introducing positive discrimination in predictive models. In B. Custers, T. Calders, B. Schermer, and T. Zarsky, editors, *Discrimination and Privacy in the Information Society*, chapter 14, pages 255–270. Springer Berlin Heidelberg, 2013.

[296] J. Vinagre, A. M. Jorge, and J. Gama. Evaluation of recommender systems in streaming environments. In *ACM RecSys Workshop on Recommender Systems Evaluation: Dimensions and Design*, REDD '14. ACM, 2014.

[297] J. Vinagre, A. M. Jorge, and J. Gama. Fast incremental matrix factorization for recommendation with positive-only feedback. In V. Dimitrova, T. Kuflik, D. Chin, F. Ricci, P. Dolog, and G.-J. Houben, editors, *User Modeling, Adaptation, and Personalization*, pages 459–470. Springer International Publishing, 2014.

[298] J. Vinagre, A. M. Jorge, and J. Gama. Online bagging for recommender systems. *Expert Systems*, 35(4):e12303, 2018.

[299] J. Vinagre, A. Mário Jorge, and J. Gama. Online gradient boosting for incremental recommender systems. In L. Soldatova, J. Vanschoren, G. Papadopoulos, and M. Ceci, editors, *Discovery Science*, pages 209–223. Springer International Publishing, 2018.

[300] E. G. Vozalis and K. G. Margaritis. Recommender systems: An experimental comparison of two filtering algorithms. In *Proc. of the Ninth Panhellenic Conference in Informatics*, 2003.

[301] W. Wallach and C. Allen. Moral machines: teaching robots right from wrong. *Ethics and Information Technology*, 12(4):359–361, Dec 2010.

[302] H. Wang, Q. Wu, and H. Wang. Factorization bandits for interactive recommendation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pages 2695–2702. AAAI Press, 2017.

[303] J. Wang. *Session aware recommender system in e-commerce*. PhD thesis, University of California, 2013.

[304] S. Wang, M. Gong, H. Li, and J. Yang. Multi-objective optimization for long tail recommendation. *Knowledge-Based Systems*, 104:145 – 155, 2016.

[305] X. Wang, Y. Wang, D. Hsu, and Y. Wang. Exploration in interactive personalized music recommendation: A reinforcement learning approach. *ACM Trans. Multimedia Comput. Commun. Appl.*, 11(1):7:1–7:22, 2014.

[306] Y. Wang and A. Kobsa. Respecting users' individual privacy constraints in web personalization. In C. Conati, K. McCoy, and G. Paliouras, ed-

itors, *User Modeling 2007: 11th International Conference*, pages 157–166. 2007.

[307] C. Wohlin. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proc. of the 18th International Conference on Evaluation and Assessment in Software Engineering*, EASE '14, pages 38:1–38:10. ACM, 2014.

[308] C. Wu, M. Yan, and L. Si. Session-aware information embedding for e-commerce product recommendation. *CoRR*, abs/1707.05955, 2017.

[309] M. Wu. Collaborative filtering via ensembles of matrix factorization. In *KDDCup 2007*, pages 43–47, 2007.

[310] L. Xiao, Z. Min, Z. Yongfeng, G. Zhaoquan, L. Yiqun, and M. Shaoping. Fairness-aware group recommendation with pareto-efficiency. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, RecSys '17, pages 107–115. ACM, 2017.

[311] Y. Xu, T. Ma, M. Tang, and T. Wei. A survey of privacy preserving data publishing using generalization and suppression. *Applied Mathematics & Information Sciences*, 8:1103–1116, May 2014.

[312] Z. Xu. The knapsack problem with a minimum filling constraint. *Naval Research Logistics*, 60(1):56–63, 2013.

[313] K. Yang and J. Stoyanovich. Measuring fairness in ranked outputs. In *Proceedings of the 29th International Conference on Scientific and Statistical Database Management*, pages 22:1–22:6. ACM, 2017.

[314] S. Yao and B. Huang. New fairness metrics for recommendation that embrace differences. *CoRR*, abs/1706.09838, 2017.

[315] G.-E. Yap, X.-L. Li, and P. S. Yu. Effective next-items recommendation via personalized sequential pattern mining. In *Proceedings of the 17th International Conference on Database Systems for Advanced Applications - Volume Part II*, DASFAAâĂŹ12, page 48âĂŞ64. Springer-Verlag, 2012.

[316] H. Yu, L. Litchfield, T. Kernreiter, S. Jolly, and K. Hempstalk. Complementary recommendations: A brief survey. In *Proc. of International Conference on High Performance Big Data and Intelligent Systems, HPBD and IS 2019*, pages 73–78, 2019.

[317] H. Yu and M. O. Riedl. A sequential recommendation approach for interactive personalized story generation. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Vol-*

ume 1, AAMAS âĂŹ12, page 71âĂŞ78. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[318] M. J. Zaki. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 12(3):372–390, May 2000.

[319] M. Zehlike, F. Bonchi, C. Castillo, S. Hajian, M. Megahed, and R. Baeza-Yates. Fa*ir: A fair top-k ranking algorithm. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 1569–1578. ACM, 2017.

[320] J. Zhan, C.-L. Hsieh, I.-C. Wang, T.-S. Hsu, C.-J. Liau, and D.-W. Wang. Privacy-preserving collaborative recommender systems. *Trans. Sys. Man Cyber Part C*, 40(4):472–476, July 2010.

[321] L. Zhang, Y. Wu, and X. Wu. A causal framework for discovering and removing direct and indirect discrimination. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, pages 3929–3935. AAAI Press, 2017.

[322] W. Zhang, H. Sun, X. Liu, Xiaohui, and Guo. An incremental tensor factorization approach for web service recommendation. In *2014 IEEE International Conference on Data Mining Workshop*, pages 346–351, Dec 2014.

[323] G. Zhao, M. L. Lee, W. Hsu, and W. Chen. Increasing temporal diversity with purchase intervals. In *Proc. of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 165–174. ACM, 2012.

[324] G. Zhao, M. L. Lee, and H. Wynne. Utilizing purchase intervals in latent clusters for product recommendation. In *Proc. of the 8th Workshop on Social Network Mining and Analysis*, SNAKDD'14, pages 4:1–4:9. ACM, 2014.

[325] Q. Zhao, Y. Zhang, D. Friedman, and F. Tan. E-commerce recommendation with personalized promotion. In *Proc. of the 9th ACM Conference on Recommender Systems*, RecSys '15, pages 219–226. ACM, 2015.

[326] X. Zhao, W. Zhang, and J. Wang. Interactive collaborative filtering. In *Proc. of the 22nd ACM international conference on Conference on Information and Knowledge Management*, CIKM '13, pages 1411–1420. ACM, 2013.

[327] Y. Zheng, T. Dave, N. Mishra, and H. Kumar. Fairness in reciprocal recommendations: A speed-dating study. In *Adjunct Publication of the 26th Conference on User Modeling, Adaptation and Personalization*, UMAP '18, pages 29–34. ACM, 2018.

[328] Z. Zhu, X. Hu, and J. Caverlee. Fairness-aware tensor-based recommendation. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, pages 1153–1162. ACM, 2018.

ISBN 978-91-7877-074-8 (print)
ISBN 978-91-7877-075-5 (pdf)

MALMÖ UNIVERSITY
205 06 MALMÖ, SWEDEN
WWW.MAU.SE