Towards Responsible AI in Recommender Systems

Department of Electrical and Information Engineering

ELECTRICAL AND INFORMATION ENGINEERING PH.D. PROGRAM
SSD: ING-INF/05– INFORMATION PROCESSING SYSTEMS

Final Dissertation

Towards Responsible AI in Recommender Systems

by

Claudio Pomo

Supervisors

Prof. Tommaso Di Noia

Prof. Francesco Maria Donini

Coordinator of Ph.D. Program

Prof. Mario Carpentieri

Course n°35, 01/11/2019-31/10/2022

Abstract

One of the most interesting success stories of Artificial Intelligence (AI) is the employment of recommender systems (RSs) by companies like Netflix, YouTube, and Amazon. The recommendation systems have evolved over time due to ongoing research in this field, and they are now frequently capable of making astoundingly effective suggestions. One could assume that the recommendation dilemma is almost resolved, given the sheer number of articles published each year. However, there are new challenges that this field has been facing in recent years. Specifically, it involves facing all the sociological, cognitive, and legislative challenges that have come to the fore in the last period. Hot topics in this area of interest are the quality of new and old models in terms of accuracy and the dimensions, such as exposure to the bias of specific groups of items or users. Also of great importance are the transparency characteristics of the models for the suggestions they propose: the model must be interpretable and capable of explaining the recommendation given to the user. How can we independently evaluate the performance of models for recommender systems? Is there a benchmarking base against which we can compare? What metrics need to be considered in a fair recommendation context? Are there metrics suitable for measuring the quality of an explanation? What techniques are best suited to address this challenge of transparency to the consumer?

This dissertation intends to outline a path about the issues of responsible AI in the context of recommender systems, starting with the problems about reproducibility and benchmarking and then addressing the issue of model performance with respect to metrics beyond accuracy and how this analysis is critically important in the context of consumer experience. Finally, the topic of explainable recommendation techniques and what impact these have in terms of performance and user experience will be addressed. Starting from the recent academic literature in the area, this thesis will intertwine with the issues close to the field of responsible AI by offering insights in the following directions: (i) we propose an unambiguous framework for recommender systems that is the core of common approach for benchmarking; (ii) we measure and compare various collaborative filtering models for recommender systems in terms of

accuracy- and over-accuracy-based metrics and their trade-offs; (iii) we improve user experience through non-trivial and explainable recommendations.

Publications

Some ideas and figures have appeared previously in other publications. A complete list of my publications is available in the following (the symbol * denotes papers where I contributed as main author).

[1] Vito Walter Anelli, Luca Belli, Yashar Deldjoo, Tommaso Di Noia, Antonio Ferrara, Fedelucio Narducci, and Claudio Pomo. "Pursuing Privacy in Recommender Systems: the View of Users and Researchers from Regulations to Applications." In: *RecSys '21: Fifteenth ACM Conference on Recommender Systems, Amsterdam, The Netherlands, 27 September 2021 - 1 October 2021*. Ed. by Humberto Jesús Corona Pampıín, Martha A. Larson, Martijn C. Willemsen, Joseph A. Konstan, Julian J. McAuley, Jean Garcia-Gathright, Bouke Huurnink, and Even Oldridge. ACM, 2021, pp. 838–841. DOI: 10.1145/3460231.3473326.

[2] *Vito Walter Anelli, Alejandro Bellogıín, Antonio Ferrara, Daniele Malitesta, Felice Antonio Merra, Claudio Pomo, Francesco M. Donini, Eugenio Di Sciascio, and Tommaso Di Noia. "The Challenging Reproducibility Task in Recommender Systems Research between Traditional and Deep Learning Models." In: *Proceedings of the 30th Italian Symposium on Advanced Database Systems, SEBD 2022, Tirrenia (PI), Italy, June 19-22, 2022*. Ed. by Giuseppe Amato, Valentina Bartalesi, Devis Bianchini, Claudio Gennaro, and Riccardo Torlone. Vol. 3194. CEUR Workshop Proceedings. CEUR-WS.org, 2022, pp. 514–521.

[3] *Vito Walter Anelli, Alejandro Bellogıín, Antonio Ferrara, Daniele Malitesta, Felice Antonio Merra, Claudio Pomo, Francesco Maria Donini, and Tommaso Di Noia. "Elliot: A Comprehensive and Rigorous Framework for Reproducible Recommender Systems Evaluation." In: *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*. Ed. by Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai. ACM, 2021, pp. 2405–2414. DOI: 10.1145/3404835.3463245.

[4] Vito Walter Anelli, Alejandro Bellogín, Antonio Ferrara, Daniele Malitesta, Felice Antonio Merra, Claudio Pomo, Francesco Maria Donini, and Tommaso Di Noia. "V-Elliot: Design, Evaluate and Tune Visual Recommender Systems." In: *RecSys '21: Fifteenth ACM Conference on Recommender Systems, Amsterdam, The Netherlands, 27 September 2021 - 1 October 2021*. Ed. by Humberto Jesús Corona Pampín, Martha A. Larson, Martijn C. Willemsen, Joseph A. Konstan, Julian J. McAuley, Jean Garcia-Gathright, Bouke Huurnink, and Even Oldridge. ACM, 2021, pp. 768–771. DOI: 10.1145/3460231.3478881.

[5] *Vito Walter Anelli, Alejandro Bellogín, Antonio Ferrara, Daniele Malitesta, Felice Antonio Merra, Claudio Pomo, Francesco Maria Donini, Eugenio Di Sciascio, and Tommaso Di Noia. "How to Perform Reproducible Experiments in the ELLIOT Recommendation Framework: Data Processing, Model Selection, and Performance Evaluation." In: *Proceedings of the 11th Italian Information Retrieval Workshop 2021, Bari, Italy, September 13-15, 2021*. Ed. by Vito Walter Anelli, Tommaso Di Noia, Nicola Ferro, and Fedelucio Narducci. Vol. 2947. CEUR Workshop Proceedings. CEUR-WS.org, 2021.

[6] *Vito Walter Anelli, Alejandro Bellogín, Tommaso Di Noia, Francesco Maria Donini, Vincenzo Paparella, and Claudio Pomo. "Adherence and Constancy in LIME-RS Explanations for Recommendation (Long paper)." In: *Joint Workshop Proceedings of the 3rd Edition of Knowledge-aware and Conversational Recommender Systems (KaRS) and the 5th Edition of Recommendation in Complex Environments (ComplexRec) co-located with 15th ACM Conference on Recommender Systems (RecSys 2021), Virtual Event, Amsterdam, The Netherlands, September 25, 2021*. Ed. by Vito Walter Anelli et al. Vol. 2960. CEUR Workshop Proceedings. CEUR-WS.org, 2021.

[7] *Vito Walter Anelli, Alejandro Bellogín, Tommaso Di Noia, Francesco Maria Donini, Vincenzo Paparella, and Claudio Pomo. "An Analysis of Local Explanation with LIME-RS." In: *Proceedings of the 12th Italian Information Retrieval Workshop 2022, Milan, Italy, June 29-30, 2022*. Ed. by Gabriella Pasi, Paolo Cremonesi, Salvatore Orlando, Markus Zanker, David Massimo, and Gloria Turati. Vol. 3177. CEUR Workshop Proceedings. CEUR-WS.org, 2022.

[8] *Vito Walter Anelli, Alejandro Bellogín, Tommaso Di Noia, Dietmar Jannach, and Claudio Pomo. "Top-N Recommendation Algorithms: A Quest for the State-of-the-Art." In: *UMAP '22: 30th ACM Conference on User Modeling, Adaptation and Personalization, Barcelona, Spain, July 4 - 7, 2022*. Ed. by Alejandro Bellogín, Ludovico Boratto, Olga C. Santos, Liliana Ardissono, and Bart Knijnenburg. ACM, 2022, pp. 121–131. DOI: 10.1145/3503252.3531292.

[9] *Vito Walter Anelli, Alejandro Bellogıín, Tommaso Di Noia, and Claudio Pomo. "Reenvisioning the comparison between Neural Collaborative Filtering and Matrix Factorization." In: *RecSys '21: Fifteenth ACM Conference on Recommender Systems, Amsterdam, The Netherlands, 27 September 2021 - 1 October 2021*. Ed. by Humberto Jesús Corona Pampín, Martha A. Larson, Martijn C. Willemsen, Joseph A. Konstan, Julian J. McAuley, Jean Garcia-Gathright, Bouke Huurnink, and Even Oldridge. ACM, 2021, pp. 521–529. DOI: 10.1145/3460231.3475944.

[10] *Vito Walter Anelli, Yashar Deldjoo, Tommaso Di Noia, Eugenio Di Sciascio, Antonio Ferrara, Daniele Malitesta, and Claudio Pomo. "Reshaping Graph Recommendation with Edge Graph Collaborative Filtering and Customer Reviews." In: *Workshop Deep Learning for Search and Recommendation*. 2022.

[11] *Vito Walter Anelli, Yashar Deldjoo, Tommaso Di Noia, Eugenio Di Sciascio, Antonio Ferrara, Daniele Malitesta, and Claudio Pomo. "How Neighborhood Exploration influences Novelty and Diversity in Graph Collaborative Filtering." In: *MORS@RecSys*. Vol. 3268. CEUR Workshop Proceedings. CEUR-WS.org, 2022.

[12] Carmelo Ardito, Tommaso Di Noia, Eugenio Di Sciascio, Domenico Lofù, Giulio Mallardi, Claudio Pomo, and Felice Vitulano. "Towards a Trustworthy Patient Home-Care Thanks to an Edge-Node Infrastructure." In: *Human-Centered Software Engineering - 8th IFIP WG 13.2 International Working Conference, HCSE 2020, Eindhoven, The Netherlands, November 30 - December 2, 2020, Proceedings*. Ed. by Regina Bernhaupt, Carmelo Ardito, and Stefan Sauer. Vol. 12481. Lecture Notes in Computer Science. Springer, 2020, pp. 181–189. DOI: 10.1007/978-3-030-64266-2_11.

[13] Vito Bellini, Giovanni Maria Biancofiore, Tommaso Di Noia, Eugenio Di Sciascio, Fedelucio Narducci, and Claudio Pomo. "GUapp: A Conversational Agent for Job Recommendation for the Italian Public Administration." In: *2020 IEEE Conference on Evolving and Adaptive Intelligent Systems, EAIS 2020, Bari, Italy, May 27-29, 2020*. IEEE, 2020, pp. 1–7. DOI: 10.1109/EAIS48028.2020.9122756.

[14] Giandomenico Cornacchia, Vito Walter Anelli, Giovanni Maria Biancofiore, Fedelucio Narducci, Claudio Pomo, Azzurra Ragone, and Eugenio Di Sciascio. "Auditing fairness under unawareness through counterfactual reasoning." In: *Information Processing & Management* 60.2 (2023), p. 103224.

[15] *Giandomenico Cornacchia, Francesco M. Donini, Fedelucio Narducci, Claudio Pomo, and Azzurra Ragone. "Explanation in Multi-Stakeholder Recommendation for Enterprise Decision Support Systems." In: *Advanced Information Systems Engineering Workshops - CAiSE 2021 International Workshops, Melbourne, VIC, Australia, June 28 - July 2, 2021, Proceedings*. Ed. by Artem Polyvyanyy and Stefanie Rinderle-Ma.

Vol. 423. Lecture Notes in Business Information Processing. Springer, 2021, pp. 39–47. DOI: 10.1007/978-3-030-79022-6_4.

[16] Tommaso Di Noia, Francesco Maria Donini, Dietmar Jannach, Fedelucio Narducci, and Claudio Pomo. "Conversational recommendation: Theoretical model and complexity analysis." In: *Inf. Sci.* 614 (2022), pp. 325–347.

[17] Juri Di Rocco, Davide Di Ruscio, Claudio Di Sipio, Phuong Thanh Nguyen, and Claudio Pomo. "On the Need for a Body of Knowledge on Recommender Systems (Short paper)." In: *Joint Workshop Proceedings of the 3rd Edition of Knowledge-aware and Conversational Recommender Systems (KaRS) and the 5th Edition of Recommendation in Complex Environments (ComplexRec) co-located with 15th ACM Conference on Recommender Systems (RecSys 2021), Virtual Event, Amsterdam, The Netherlands, September 25, 2021*. Ed. by Vito Walter Anelli et al. Vol. 2960. CEUR Workshop Proceedings. CEUR-WS.org, 2021.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

The enormous amount of data on the World Wide Web (WWW), which has grown tremendously over the past three decades, has brought users closer to each other and to previously undiscovered goods [159], services [64], music [103], and films [37]. The problem is that it gets harder to find the best content in this gold mine when more information is available. Users are frequently confronted with a plethora of options, making it challenging for them to select one, especially when they have a finite amount of money or other resources to work with. For instance, a music search engine may still return thousands of songs if you type in "classic rock".

In these circumstances, Recommender Systems (RSs) can greatly aid users in making decisions since they direct them toward stuffthey have not yet tried but are likely to find relevant and interesting. One example is the return of a ranked list of items [209]. Modern RSs typically offer individualized and personalized recommendations to various individuals or user groups. They offer suggestions based on several presumptions in their formulation, such as the possibility that users can establish their choices based on their preferences or the actions of other users who share those preferences. As a result, recommender systems increase competition between online services and brick-and-mortar retailers, improving user experience, provider revenue, and product variety. Despite the enormous advances in the RS field, the bulk of current research works have concentrated on improving system accuracy, with just a small portion of the literature also taking other significant RS qualities and values, such as diversity and novelty, into account [51]. Although recommender systems represent a bigger sociotechnical system, the ranking algorithm is still just one component of it. Given that algorithm research predominates in the field today, it becomes clear that our research community seems to concentrate too much on a single aspect of the issue, particularly one where the use of increasingly complicated algorithms may only produce

declining results [127]. In fact, end consumers have not been adequately taken into account by current accuracy-oriented RSs in the context of human-machine interaction as a whole, where other significant elements are frequently seen as crucial for the whole user experience [138].

Certain extra factors need special consideration. First, the internet in which RSs are used is getting more complicated and full of hazards from a variety of sources and types, such as cyberattacks, noisy and false information, and system bias. As a result, there is a demand for trustworthy RSs (TRSs) that compete in the difficult and complicated online world. Second, there is an increased pressure on RSs from stakeholders like users, owners, and regulators. These stakeholders need recommendations that are trustworthy, which includes things like robustness, fairness, explainability, privacy preservation, etc. in addition to accuracy. In fact, in some vital and sensitive industries, like as finance and health, where extremely reliable RSs are required, trustworthiness is even more crucial than correctness. Accuracy shouldn't be an RS's primary goal; trustworthiness must come first, it has been agreed upon both in academia and the business world. A new RS paradigm, or TRSs, is urgently needed as a result of these investigations [259].

The term "trustworthy" refers to an entity that a subject may depend on to be decent, honest, and sincere[1]. Among the specific characteristics of "trustworthy" are reliability, dependability, faithfulness, honor, creditworthiness, and responsibility [284]. An RS that its shareholder may trust to be good, trustworthy, reliable, dependable, and faithful is referred to as a TRS in general.

1.1 Thesis Statement

The chapters of this thesis are as self-contained as possible and present the notions of specific problems, architectures, paradigms, data structures, and metrics related with their content. Moreover, each chapter independently surveys the state-of-the-art of the specific problem it deals with, showing the most interesting solutions and their limitations in the literature of that field.

In the next chapter, we propose a brief but comprehensive introduction to the most important recurring topics of this dissertation. In detail, we analyze the recommendation problem with its models, solutions, and evaluation techniques (Chapter 2).

From this overview, we look at the problem of reproducibility and how an entire evaluation pipeline can be tested consistently against the different recommendation

[1]https://www.oxfordlearnersdictionaries.com/definition/american_english/trustworthy

models involved in a specific analysis. Next, we show how a careful investigation of reproducibility allows for consistent exploration of the dimensions of novelty, diversity, and fairness (e.g., beyond accuracy metrics) in an offline evolution scenario for RSs. These kinds of metrics have a significant impact on the user experience of the customer using a generic RS. For this reason, another aspect strongly related to this dimension is analyzed: the possible explanations associated with recommendations. Specifically, the focus is analyzing explainable recommendations generated by post-hoc approaches. In the end, a formal model is proposed for generating counterfactual explanations for Decision Support Systems (DSSs) in an enterprise scenario.

1.2 Research Contributions

This thesis investigate the aspects of TRSs peculiar to the strand of responsible AI: reproducibility of results, fair and explainable recommendations. Each of these issues is crucial in the TRSs research area, and this thesis connects these aspects by investigating their distinctiveness in the context of personalization of information access. The following sections provide additional details on this thesis's research goals and contributions. The author of this thesis, **Claudio Pomo** is the **main and corresponding author** of the scientific publications related to the research contributions presented in this dissertation[2].

1.2.1 Ch. 3: Building a rigorous and robust framework for RSs evaluation.

Contributions. In Chapter 3 of this thesis, we present our experiment management framework for a recommendation scenario. Introducing an analysis about the problem of reproducibility in RSs, we will see how this has led to the emergence of some state-of-the-art model testing/evaluation frameworks for the current research area. This investigation prompted us to release an all-encompassing testing framework for RSs, from dataset management and partitioning to rigorous model evaluation, without neglecting hyperparameter optimization. The result flowed into the opensource software Elliot, currently the only tool in the RS landscape that offers the ability to manage an entire pipeline of recommendations in a rigorous and transparent manner, leveraging a single, appropriately packaged configuration file.

[2]The authors of the publications are alphabetically ordered. The corresponding author of each publication is reported in the original articles.

Publications. The results of this effort resulted in the publication of a scientific paper *"Elliot: a comprehensive and rigorous framework for reproducible recommender systems evaluation"* [15] presented at the 44th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR) 2021. A shortened version of this work was the subject of presentation at the 11th edition of the Italian Information Retrieval Workshop (IIR) 2021 and published in the discussion paper *"How to Perform Reproducible Experiments in the ELLIOT Recommendation Framework: Data Processing, Model Selection, and Performance Evaluation"* [16]. In addition, the paper *"The Challenging Reproducibility Task in Recommender Systems Research between Traditional and Deep Learning Models"* [13], presented at the 30th Italian Symposium on Advanced Database Systems (SEBD) 2022, highlights the critical issues of RSs when it comes to reproducibility.

Role of Ph.D. Candidate. Corresponding author of all the above-mentioned articles [13, 15, 16].

1.2.2 Ch. 4: Reproducibility and Repricability studies spanning classical Collaborative Filtering and beyond.

Contributions. In this thesis chapter, we will analyze in detail the problem of reproducibility of results in the context of Collaborative Filtering (CF) type RSs. Starting from the scientific literature and its experimental results, we will show how, despite the community's efforts to make RSs perform better in terms of accuracy, some recent works add nothing, and some historical research area models still perform well if the evaluation scenario found to be consistent regarding experimental choices. Our analysis of CF models continues beyond accuracy metrics. Still, it takes the opportunity to showcase how we have often chased the path of improving accuracy performance while sidelining research for novelty/diversity or bias/fairness phenomena that strongly impact the user experience.

Publications. The Elliot framework has also proven to be effectively useful in the context of reproducibility and replication of results; in fact, two papers were developed with its support in this context. The first was presented at the 15th ACM Conference on Recommender Systems (RecSys) 2021 and is entitled *"Reenvisioning the comparison between neural collaborative filtering and matrix factorization"* [20], while a second paper entitled *"Top-n recommendation algorithms: A quest for the state-of-the-art"* [19] was presented at the 30th ACM Conference on User Modeling, Adaptation and Personalization (UMAP) 2022.

Role of Ph.D. Candidate. Corresponding author of all the above-mentioned articles [19, 20]

1.2.3 Ch. 5: Analysis of the impact of beyond accuracy metrics in modern recommender systems.

Contributions. The analysis conducted with the previously mentioned tool, *"Reenvisioning the comparison between neural collaborative filtering and matrix factorization"*, found that state-based collaborative recommendation models achieve remarkable results in terms of accuracy but fail to perform as well when measuring the frequency of suggesting less popular items. This observation motivated the analysis of a type of RSs belonging to the CF family, that is inspired by Graph Learning techniques (GLRSs). This new approach has shown great promise in terms of accuracy. Still, no rigorous evaluation revealed the performance in terms of novelty/diversity and bias/fairness of the recommendations produced. The analysis involved the detailed study of GLRSs and their information propagation (message passing) characteristics through the various nodes. In addition, a new approach called Edge Graph Collaborative Filtering (EGCF) was proposed and showed significant results in terms of both accuracy and novelty/diversity. Moreover, an obvious trade-offbetween accuracy and user experience metrics (novelty/diversity and bias/fairness) motivated a new development in this line of research. To close this gap of this analysis between GLRSs and classical RSs belonging to the collaborative filtering (CF) family, a new approach of rigorous analysis based on a multi-objective approach guided by Pareto frontiers is introduced.

Publications. The research contributions presented in this chapter are based on two works. The first one is presented at the 2nd Workshop on Multi-Objective Recommender Systems held in conjunction with the 16th ACM Recommender Systems Conference (RecSys) 2022 and titled *"How Neighborhood Exploration influences Novelty and Diversity in Graph Collaborative Filtering"* [21]. This paper represents the first attempt to analyze the dimensions of novelty and diversity concerning message passing techniques typical of GLRSs. The evidence from this work motivated another contribution entitled *"Reshaping Graph Recommendation with Edge Graph Collaborative Filtering and Customer Reviews"* [22] that was presented at the Workshop Deep Learning for Search and Recommendation held in conjunction with the 31st ACM International Conference on Information and Knowledge Management (CIKM) 2022. Furthermore, a research paper about the introduction of a new off-line evaluation approach to measure the severity of the trade-offbetween accuracy and bias/fairness

metrics is currently under review as a long paper entitled *"Auditing Consumer- and Producer-Fairness in Graph Collaborative Filtering"*.

Role of Ph.D. Candidate. Corresponding author of the articles [21, 22] and the research contribution presented in a paper under review. These elements are presented in Chapter 4.

1.2.4 Ch. 6: Impact of explainable recommendations on the user experience.

Contributions. One of the crucial issues for TRSs is definitely that of explanations. Research contributions in this direction have become attractive again due to the renewed interest in eXplainable Artificial Intelligence (XAI). In order to improve the effectiveness, efficiency, persuasiveness, and user satisfaction of recommender systems, explainable recommendation refers to the personalized recommendation algorithms that address the problem of why – they not only give the user the suggestions but also make the user aware of why such items are recommended by generating recommendation explanations. We focused on post-hoc approaches, particularly the LIME-RS technique. Our analysis aimed to demonstrate how often this type of explanation based on local surrogate models does not prove consistent with user and item characteristics, risking producing ineffective explanations. Motivated by these results, we proposed a formal approach for generating explanations from Multi-Stakeholder type RSs (MS-RSs). In this context, we considered the point of view of counterfactual explanations. We highlighted the pros and cons of their application in a scenario that explained the recommendation for the consumer and the policy adopted by the system for the considered stakeholder.

Publications. Post-hoc approaches being part of the established literature in both XAI and explainable recommendation, the first paper analyzed the reliability of explanations based on local surrogate models. It was presented at the 3rd Knowledge-aware and Conversational Recommender Systems (KaRS) workshop held in conjunction with the 15th ACM Recommender Systems Conference (RecSys) 2021 titled *"Adherence and Constancy in LIME-RS Explanations for Recommendation"* [17]. The second contribution of this analysis was proposed as a discussion for an extended abstract at the 12th edition of the Italian Information Retrieval Workshop (IIR) 2022 entitled *"An Analysis of Local Explanation with LIME-RS"* [18]. The proposed model for generating counterfactual explanations in a Multi-Stakeholder context was presented at the Advanced Information Systems Engineering Workshops held in conjunction

with the 33rd International Conference on Advanced Information Systems Engineering (CAiSE) 2021 under the title *"Explanation in multi-stakeholder recommendation for enterprise decision support systems"* [63].

Role of Ph.D. Candidate. Corresponding author of all the above-mentioned articles [17, 18, 63].

1.3 Bibliographical Notes

This section describes in depth the research articles published during the Ph.D. but not discussed in the dissertation. Indeed, the following works have been conducted as simultaneous topics whose research questions have been raised while studying the literature.

Two papers were proposed for the topic regarding conversational RSs, motivated by the possibility of generating explanations with respect to proposed recommendations due to preferences elicited during user dialogue. The first paper has a more applied character and was presented as a contribution to the IEEE Workshop on Evolving and Adaptive Intelligent Systems (EAIS) 2020 with the title *"GUapp: A Conversational Agent for Job Recommendation for the Italian Public Administration"* [31]. In this paper, we present GUapp, a chatbot that aims to allow users to interact with the app through natural language. Thanks to that, the search and recommendation process becomes incremental, and the user can add new requirements at each stage of the interaction. The second paper proposes a complexity analysis algorithm for CRSs. In this research work, a dimension that has never been considered in this context was analyzed to evaluate a priori the goodness of a dialogue strategy for a RS of this type. This paper was published in the journal Information Science titled *"Conversational Recommendation: Theoretical Model and Complexity Analysis"* [75].

Regarding the line of research on reproducibility, in addition to the already mentioned works, the following was presented a demonstration paper fully dedicated to the integration of visual-based recommenders has been presented at RecSys 2021 in the indexed article named "*V-Elliot: Design, Evaluate and Tune Visual Recommender Systems*" [14]. In addition, a paper titled *"On the Need for a Body of Knowledge on Recommender Systems"* [213] was presented at the 3rd Knowledge-aware and Conversational Recommender Systems (KaRS) workshop held in conjunction with the 15th ACM Recommender Systems Conference (RecSys) 2021 regarding a software formalization of the pipeline for developing and evaluating a recommender model. This contribution was made in collaboration with the research group led by Davide Di

Ruscio at Aquila University. Completing the picture of this line of research is the contribution for the analysis of state-of-the-art models in RSs in the *privacy preserving* context. This effort resulted in the presentation of a hands-on tutorial presented at 15th ACM Recommender Systems Conference (RecSys) 2021 entitled *"Pursuing Privacy in Recommender Systems: the View of Users and Researchers from Regulations to Applications"* [12].

Finally, I co-authored a paper entitled "FastSHAP explanation for Recommender Systems" with David Wardrope and Brian Mohr. This work, developed during my internship at Amazon.com, is currently under review. The contributions in this paper aim to show the applicability of a *Shapely Values* approximation approach in a large-scale recommendation context, and some critical user experience scenarios in specific user modeling contexts are identified.

Chapter 2

Recommender Systems

Recommender systems can deliver recommendations to users when faced with an extensive catalog of goods to choose from or whenever they want to obtain suggestions. The users of recommender systems are modeled by learning from their previous behavioral data (such as movies they have watched, ratings they have given, items they have purchased, and websites they have visited), and they are then directed toward newly discovered objects that are most likely to be of interest based on various assumptions.

This chapter will first give a formal definition of the recommendation problem. The pioneer models and architectures will be reviewed along with their taxonomy next. Finally, we will provide an overview of performance evaluation and discuss the measures that were employed in this dissertation in detail.

2.1 Definition

In a more formal sense, we will refer to the set of users in the system as $\mathcal{U}$, and the set of items as $\mathcal{I}$ (e.g., the catalog of an online shop). The user-item preference matrix may then be denoted as $\mathbf{R} \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{I}|}$. This matrix may contain either explicit feedback or implicit feedback. In the first scenario, we can have, for example, $r_{ui} \in \{\emptyset\} \cup \{\ 1, \ldots, 5\}$ if the users are asked to leave a rating of up to 5 stars for a product, with $\emptyset$ denoting a missing rating. This would be the case if the users were asked to leave a rating for a product. In the case of implicit feedback, for instance, we are able to observe data in the form of $r_{ui} \in \{\emptyset\} \cup \{\ 1\}$ with $r_{ui} = 1$ if u interacted with i (for example, the user visited a webpage, clicked on a video, or purchased a product) and $r_{ui} = \emptyset$ or in the form of $r_{ui} \in \{\emptyset\} \cup \{\ \mathit{like}(1), \mathit{dislike}(0)\}$ if not. The items in a catalog typically count in the thousands or millions, and as a result, most users will naturally only rate or enjoy

a small percentage of those items. In the majority of cases, the percentage of ratings that are missing from well-known datasets used in recommendation research (such as MovieLens and Netflix, for example) is greater than 95%. When a user is prompted to provide explicit ratings, the datasets frequently display a large skew that favors values that are either extremely high or extremely low. Notably, we define the set of items that user u interacted with as $\mathcal{I}_u$, which stands for "interacted with by user", i.e.,

$$\mathcal{I}_u = \{i \in \mathcal{I} \mid r_{ui} \neq \emptyset\}. \tag{2.1}$$

In general, recommendation tasks are associated with a diverse set of activities, such as finding all of the possible good items for a user or recommending an ordered sequence or a bundle of items. On the other hand, *rating prediction* and *top-k recommendation* [66, 183] are the two tasks that have received the greatest attention and research in the field of literature.

Definition 1 (Rating prediction task)**.** *Given a user $u \in \mathcal{U}$ and an item $i \in \mathcal{I} \setminus \mathcal{I}_u$ not rated by u, the rating prediction task aims to predict the missing rating of u to i. Formally, the goal is to learn a function $f : \mathcal{U} \times \mathcal{I} \to \mathbb{R}$, often defined as a regression or a classification.*

Definition 2 (Top-k recommendation task)**.** *Given a user $u \in \mathcal{U}$, the top-k recommendation task aims to provide u with a list containing k items from $\mathcal{I} \setminus \mathcal{I}_u$ most likely to interest u, ordered by a proper utility function $s : \mathcal{U} \times \mathcal{I} \to \mathbb{R}$.*

Often, the top-k recommendation task is solved by ranking the items of $\mathcal{I}$ based on the predictions given by f, which acts as a utility function.

The solution to the above-mentioned recommendation problems heavily depends on the selected utility function s or the prediction function f — usually, but not necessarily, a machine learning model — and on the type of information encoded in the data.

While the idea of utility may be often exclusively associated with user satisfaction, it actually includes the revenue of content creators and content providers (e.g., depending on the number and the diversity of the items sold), and the users' loyalty to the service. The prediction of the utility, or at least the comparison of the utility of some items, is the core function of recommender systems for understanding whether an item is worth recommending [210].

In the following, we present a taxonomy of the most common recommendation approaches, along with their standard and simplest formulations, usually aiming to predict and estimate the value of the missing ratings.

2.2 Overview of Recommendation Approaches

Recommender systems solve the recommendation problem previously defined by relying on different assumptions. For instance, they may assume that users can make decisions about new items based on the similarity with items already consumed. Alternatively, they may assume that users' decisions may rely on the behavior of similar users.

The most common recommendation strategies can be classified into collaborative filtering, content-based filtering, and hybrid models [183]. In the following, we will present a brief but comprehensive overview of recommendation methods along with their taxonomy.

2.2.1 Collaborative Filtering Approaches

The core idea of collaborative filtering recommender system is that the rating of a user for a new item is likely to be similar to that of another user who has rated other items in a similar way [140]. Moreover, these systems assume that a user is likely to similarly rate two items if other users have given similar ratings to these two items. The methods using collaborative filtering approaches are able to provide recommendations to users through the feedback of other users and can recommend items with very different content if other users have shown interest in all these different items. Due to their formulation, these algorithms' performance highly relies on the availability of user transactions, but they have the advantage of not needing any other data source (e.g., attributes of the items).

Collaborative filtering methods can be grouped into neighborhood- and model-based methods [140]. While the former class of methods uses the collected ratings to predict the missing ratings, the model-based approaches aim to build a predictive model able to represent latent characteristics of the users and the items in the system.

Neighborhood-based collaborative approaches

Neighborhood-based recommender systems, also known as memory-based recommenders, assume that, in the user perspective, the opinion of like-minded users is useful for evaluating the value of an item. In principle, similar users prefer similar items, and similar items are preferred by similar users.

The k-nearest-neighbors (k-NN) approach, with its user-based and item-based variants (e.g., [73, 74]), is the most used one in memory-based recommendation.

In this schema, the k-NN algorithm computes a similarity matrix $\mathbf{W} \in [0,1]^{|\mathcal{U}|x|\mathcal{U}|}$ encoding in its entries w_{ij} how similar is user i whit respect to user j. Then, the k users

with the highest similarity w_{uv} to a user u constitute the k-nearest-neighborhood $\mathcal{N}(u)$ of u, in formula $\mathcal{N}(u) = \{u \in \mathcal{U} | w_{uv}$ is the $k-th$ maximum among $w_{u1}, w_{u2}, ..., w_{un}\}$. To estimate a missing rating r_{ui}, user-based k-NN considers the subset $\mathcal{N}_i(u) \subseteq \mathcal{N}(u)$ containing the neighbors of u who have rated i. Then, considering one of the simplest formulations of the algorithm, the rating r_{ui} can be estimated as:

$$\tilde{r}_{ui} = \frac{\sum_{v \in \mathcal{N}_i(u)} w_{uv} r_{vi}}{\sum_{v \in \mathcal{N}_i(u)} |w_{uv}|} \tag{2.2}$$

Intuitively, user u asks the like-minded users in $N_i(u)$ their opinion about i, and considers it for making a decision.

With item-based k-NN, the core idea is to find similarities between items by looking at the ratings they have received. Similar to the previous formulation, once the item similarity matrix $\mathbf{W}$ has been computed, each item can be associated with a set of neighbors $\mathcal{N}(i)$. To estimate a missing rating r_{ui}, only the items in $\mathcal{N}_u(i)$ (i.e., similar to i that have been rated by u) are considered:

$$\tilde{r}_{ui} = \frac{\sum_{j \in \mathcal{N}_u(i)} w_{ij} r_{uj}}{\sum_{j \in \mathcal{N}_u(i)} |w_{ij}|} \tag{2.3}$$

The similarity between two users in user- based methods, which determines the neighbors of a user, is normally obtained by comparing the ratings made by these users on the same items. Conversely, in item-based methods, the similarity between two items is obtained by comparing the ratings they have received from different users. The computation of the similarity weights is one of the most critical aspects of building a neighborhood-based recommender system, as it can have a significant impact on both its accuracy and its performance. Among the most used approaches, it is common to find Cosine similarity, Dot similarity,Pearson correlation, Jaccard index, and Euclidean distance [140].

There are several factors to be considered when choosing between user- and item-based k-NN. In terms of recommendation performance, a small number of high-confidence neighbors is preferable to a large number of neighbors for which the similarity weights are not trustable. Thus, where the number of users is much greater than the number of items, item-based methods are preferred [86]. Conversely, user-based methods usually provide more original recommendations, which may lead users to a more satisfying experience [81]. The choice is also affected by memory and computational efficiency requirements since similarity estimation complexity grows quadratically with the number of users in user-based models and with the number of items in item-

based models [183]. More efficient approaches to digging all neighbors exist in the literature [62, 92, 247] but have not been applied to the area of RSs.

Overall, neighborhood-based collaborative approaches are intuitive and simple, produce explainable recommendations, and are efficient in prediction (once the similarity matrix has been computed), albeit they can suffer from problems like *limited coverage* of items (when users have no common ratings, they are necessarily classified as non-neighbor users, thus causing some their favorite items not to be recommended to each other) and high-performance sensitivity to the lack of available data.

Model-Based Collaborative Approaches

Model-based collaborative filtering approaches use the ratings in the matrix $\mathbf{R}$ to learn a predictive machine learning model representing latent characteristics of the users and the items in the system able to explain the ratings.

These approaches are numerous and have shown improved performance with respect to memory-based collaborative filtering algorithms, especially in the context of the million-dollar prize competition opened by Netflix in October 2006 [37], where they gain significant attention. Thanks to that competition, the research community gained access to a large-scale, industrial-strength dataset of 100 million movie ratings that encouraged the rapid development of model-based recommender systems.

Typical model-based approaches include Bayesian Clustering, Latent Semantic Analysis, Latent Dirichlet Allocation, Support Vector Machines, and neural networks [140]. However, the most popular models are the ones induced by factorization of the user-item rating matrix, thanks to their attractive accuracy and scalability. Singular Value Decomposition is one of the most established techniques for decomposing a matrix and identifying latent factors [141]. Nevertheless, the high portion of missing values leads to difficulties that force to train the model on the few available data and to use regularization for averting the risk of overfitting.

Matrix factorization aims to represent both items and users as vectors in a latent space $\mathbb{R}^f$. The latent space tries to explain ratings with latent factors, automatically inferred from user feedback, representing properties (e.g., the genre of a movie) that cannot be directly interpreted by a human being observing the decomposed matrix. The value assumed by each latent factor represents, in the user vector, her interest towards that property, while, in the item vector, the extent to which the item possesses the same property. The dot product between a user vector and an item vector will capture the user's estimated interest in that item.

Notably, given the vector $\mathbf{q}_i$ representing the item i in the latent space $\mathbb{R}^f$, and the vector $\mathbf{p}_u$ representing the user u in the same space, we estimate a missing rating as:

$$\tilde{r}_{ui} = \mu + b_u + b_i + \mathbf{q}_i^T \mathbf{p}_u, \tag{2.4}$$

where μ, b_u, and b_i are the overall rating average, the user u's rating average, and item i's rating average, respectively. These three terms constitute the baseline predictor for r_{ui}, which has a detrimental impact on collaborative filtering since feedback data usually exhibit large biases (e.g., some users have systematic tendencies to give very high or low ratings and some items to receive very high or low ratings). The vectors of all the users and items in the system can be learned with techniques like stochastic gradient descent and alternating least squares by exploiting the available ratings.

The prediction in Eq. 2.4 can be extended and used as a building block for more complex models (e.g., to take into account other types of feedback or to build time-aware factor models).

Generally, model-based approaches exhibit prediction accuracy superior to other collaborative filtering techniques [140] with a relatively memory-efficient and easy-to-train compact model. Moreover, these techniques are also convenient thanks to the possibility of injecting multiple forms of user feedback and modeling specific behaviors [139, 189, 218]. Once the rating prediction estimation has been modeled, based on the task and the assumptions, the loss function to minimize can be differently written (e.g., BPR [197]). Then, we can also choose among various training strategies for learning the model parameters (e.g., stochastic gradient descent [89] and alternating least squares [119]).

However, the linearity of MF approaches is the primary point of criticism regarding these methods. Using deep neural architectures with deep neural networks that are able to model the non-linearity in data through the use of nonlinear activation functions has recently become a popular trend in the community of recommender systems. This trend was recently popularized to address this concern. In this respect, Neural Collaborative Filtering [113] and Neural Factorization Machines [108] have been recently proposed to overcome the inability of MF to capture non-linearities. He et al. [113] proposes exploiting Artificial Neural Networks to learn the affinity between $\mathbf{p}$ and $\mathbf{q}$. Let $\Phi(\cdot)$ be the transformation function of the deep neural network defined as $\Phi : \mathbb{R}^{\dim(p)+\dim(q)} \rightarrow \mathbb{R}^d$, He et al. propose to concatenate the two embeddings and predict the score as follows:

$$\psi^{\text{MLP}}(\mathbf{p}, \mathbf{q}) := \Phi([\,\mathbf{p}, \mathbf{q}]). \tag{2.5}$$

We give more details about these models in Chapter 3, where we propose an RSs framework for rigorous evaluation experiments.

2.2.2 Content-Based Filtering Approaches

Recommender systems with content-based filtering exploit attributes of items and users to produce recommendations by matching up the characteristics of a target user with the attributes of the items [183]. Characterizing items with *content information* that can effectively feed the recommendation process is usually a non-trivial task. In most cases, the items' attributes are simple keywords that are extracted from their description or metadata, albeit an increasing number of works has moved from a *keyword*-based to a *concept*-based approach by characterizing the items with semantic information extracted from structured knowledge sources, such as Wikipedia, DBpedia, Freebase, and BabelNet.

The steps to perform a content-based recommendation include [97] i) a *content analyzer* that uses information coming from the most diverse sources to represent the items in a specific description space (e.g., vector space model); ii) a *profile learner* that collects all the users' preference data, and with a chosen strategy (e.g., probabilistic methods, relevance feedback, and k-nearest neighbors) tries to generalize this data in order to build the user profile; iii) a *filtering component* that suggests relevant items by matching the profile representations against the item descriptions.

Content-based recommender systems provide several advantages, including the possibility to explain the produced recommendation based on the attributes, and the absence of the cold-start problem for the items, since no collaborative information is needed to feed the recommender. Content-based filtering recommenders also present some shortcomings [97]: they include the cold-start problem for new users, because of the dependence of their profile on their rating history. Moreover, often these systems tend to suffer from an *overspecialization* problem (or lack of serendipity) since they tend to suggest merely items homogeneous with the ones experienced in the past [123]. Finally, another known problem is the scarse availability of structured knowledge about the domain, which would consistently improve the recommendation performance.

2.2.3 Hybrid Approaches

In some cases, the limitations of collaborative filtering methods and content-based filtering methods can be partially overcome by realizing a hybrid combination of the two approaches. Hybrid recommenders aim to improve the performance of the

system by exploiting the best of both techniques. Some ways to implement hybrid recommenders consider implementing both the approaches separately and then using an aggregation function to merge the results, or injecting content information into a collaborative-filtering method, or vice versa, or building a system exploiting both the sources of information [9].

2.2.4 Evaluation

We have already seen how many algorithms and approaches can be used to build a recommender system. Its choice depends on a large and variable number of factors, including a wide range of properties of the dataset (e.g., its sparsity, the ratio between users and items). Indeed, all these factors jointly impact different dimensions of the performance of the recommender system. All these dimensions, in turn, may enormously impact the user experience and the acceptance of both the recommendation and the whole system. This makes the evaluation of a recommender system a complex task that focuses on both users and items [98].

It is noteworthy that the evaluation of a recommender system should follow an online protocol, proposing recommendations to the users and waiting for their feedback. Practically, this is not feasible for most researchers due to the limited number of users they can involve in an experiment. To perform an offline evaluation, the set of available feedback is divided into a *training set* used for learning the utility function and a *test set* for assessing the quality of the learned function. Generally, the interactions of each user are independently split, thus generating for each of them the training set of items with an interaction $\mathcal{I}_u^{\text{train}} \subset \mathcal{I}_u$ and the test set of items with an interaction $\mathcal{I}_u^{\text{test}} \subset \mathcal{I}_u$. Depending on the strategy, a user can be provided with a recommendation list containing only items present in the test set (i.e., already liked in the past) or containing only items not present in the training set (also items without ratings in the test set could be recommended) [230]. In the former case, we refer to *rated test-items* protocol, in the latter to *all unrated items* protocol. In this dissertation, we perform offline tests on all unrated items since they reflect much better the situation in many real-world recommendation tasks [230].

In the following, we briefly survey the metrics used along with this dissertation. Hereinafter, we mainly consider the recommendation problem in the form of top-k recommendation task, which is the mainstream of recommender system research. Indeed, our evaluation is oriented to assessing the quality of the ranking rather than evaluating the prediction score. Thus, the metrics in the following are presented with

their definition specifically designed for evaluating the top-k/ranking task with the all-unrated-items protocol.

Accuracy Metrics

Accuracy metrics in top-k recommendation look for the presence of *relevant* items in the top-k elements of the recommended list, with k usually in $\{1, 5, 10, 50, 100\}$. Datasets with implicit feedback have an inherent definition of relevance (e.g., positive feedback as clicks, purchases, likes). Explicit feedback data like ratings, instead, have to be discretized into binary values according to a relevance threshold. Thus, $\mathcal{I}_u^+ \subseteq \mathcal{I}_u$ is the set of items relevant to user u, and $\mathcal{I}_u^{+,\text{test}} \subseteq \mathcal{I}_u^{\text{test}}$ contains the relevant items in the test set.

Definition 3 (Precision)**.** *For each user $u \in \mathcal{U}$, let $\mathcal{L}_u$ be a recommendation list ranked according to Eq. 2, and $\mathcal{L}_u^{(1,\dots,k)}$ its top-k. Then, the precision P@k is defined as the average, over all the users, of the proportion of items in the top-k of each user that are also relevant to the user.*

$$P@k = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{|\mathcal{L}_u^{(1,\dots,k)} \cap \mathcal{I}_u^{+,\text{test}}|}{k} \tag{2.6}$$

Intuitively, precision measures the system's ability to reject any non-relevant item in the retrieved set. Recall, instead, aims to measure the system's ability to find all the relevant items.

Definition 4 (Recall)**.** *Given the premises in Definition 3, the recall R@k is defined as the average of the proportion of relevant items that are retrieved in top-k of each user.*

$$R@k = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{|\mathcal{L}_u^{(1,\dots,k)} \cap \mathcal{I}_u^{+,\text{test}}|}{|\mathcal{I}_u^{+,\text{test}}|} \tag{2.7}$$

To offer an aggregate measure of the accuracy, precision and recall are sometimes combined with each other in the *F1@k* measure [212] computed as their harmonic mean:

$$F1@k = 2\frac{P@k \cdot R@k}{P@k + R@k} \tag{2.8}$$

Finally, a widely used accuracy metric is the *discounted cumulative gain* [128] from information retrieval, which assesses the quality of a ranking based on the position of the ranked elements.

Definition 5 (Normalized Discounted Cumulative Gain)**.** *For each user, let* $l_{u(i)} \in \mathcal{L}_u^{(1,\dots,k)}$ *the recommended item in position* $i \leq k$*, and* $v_{l_{u(i)}} = 1$ *if* $l_{u(i)} \in \mathcal{I}_u^{+,\text{test}}$*, 0 otherwise. Then, define:*

$$nDCG@k = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \sum_{i=1}^{k} \frac{2^{v_{l_{u(i)}}} - 1}{\log_2(i+1)} \tag{2.9}$$

In $nDCG@k$, the numerator represents the gain that u has from being recommended item $l_{u(i)}$, with its relative positions in the list discounted logarithmically.

Beyond-Accuracy Metrics

The presence of relevant items in the recommended list may not be sufficient for the evaluation of a recommender system. We have already mentioned that, among the other things, recommendations should be engaging for the users, should increase their trust and serendipity, and should generate revenue for content providers.

Some algorithms may provide accurate recommendations, but may polarize and homogenize users' interest towards a small portion of the items (the *short head*), i.e., the ones rated by the majority of users. This is usually caused by popularity bias since recommender systems often exacerbates and perpetuates social biases present in the rating data by reinforcing the popularity of already popular products (*rich-get-richer* effect). This behavior is often referred to as the *long tail* problem, since the unpopular items, which represent the vast majority of the items, are not recommended at all.

As a consequence, there is a need for item diversity in recommendation lists to encourage serendipity, and increase users' satisfaction and sales. In the following, we introduce three sales diversity metrics widely employed in recommender systems.

Definition 6 (Item Coverage)**.** *Given the premises in Definition 3, the item coverage computes the number of items that are shown to at least one user.*

$$IC@k = \frac{|\bigcup_{u \in \mathcal{U}} \mathcal{L}_u^{(1,\dots,k)}|}{|\mathcal{I}|} \tag{2.10}$$

A higher value of item coverage implies high diversity of the recommendation lists, which may suggest a good performance in terms of personalization with respect to the users.

The other diversity measures considered in this dissertation are Gini Index (G) and Shannon Entropy (SE), that measure the distributional inequality [51], i.e., in this context, how unequally different items are recommended to the users. A recommender

system equally recommending its items is likely providing diverse recommendations and equally favoring all the elements in the catalog.

Definition 7 (Gini Index)**.** *Given the premises in Definition 3, let* $(\mathcal{L}^k, m)$ *be a multiset, with* $\mathcal{L}^k = \bigcup_{u \in \mathcal{U}} \mathcal{L}_u^{(1,\dots,k)}$ *being the set of all the recommended items and* $m : \mathcal{L}^k \to \mathbb{N}$ *a function giving the number of times an item is recommended. Suppose that the elements* $l \in \mathcal{L}^k$ *are in ascending order by the value of* $m(l)$ *and* $l^{(i)}$ *denotes the item in the* i*-th position. Then, the Gini Index is defined as:*

$$\hat{G}@k = \frac{1}{|\mathcal{L}^k| - 1} \frac{\sum_{i=1}^{|\mathcal{L}^k|} (2i - |\mathcal{L}^k| - 1) m(l^{(i)})}{\sum_{i=1}^{|\mathcal{L}^k|} m(l^{(i)})} \tag{2.11}$$

A value of $\hat{G}@k$ close to 0 reflects diversity and means that the items are equally recommended (all products have equal sales), whereas 1 represents concentration (a single item is recommended to all users). In the remainder of this dissertation, the values of Gini Index will be always provided in their *higher is better* version $G@k = 1 - \hat{G}@k$.

Definition 8 (Shannon Entropy)**.** *Given the premises in Definition 7, the Shannon entropy is defined as:*

$$SE@k = -\sum_{i=1}^{|\mathcal{L}^k|} \frac{m(l^{(i)})}{\sum_{i=1}^{|\mathcal{L}^k|} m(l^{(i)})} \log \left(\frac{m(l^{(i)})}{\sum_{i=1}^{|\mathcal{L}^k|} m(l^{(i)})} \right) \tag{2.12}$$

The entropy is 0 when a single item is always recommended, and $\log(|\mathcal{L}^k|)$ when all the items are recommended equally often.

The presence in the user's recommendation list of items that are completely foreign to the user is a way to define the concept of novelty in recommendations. There is no agreed-upon definition of what constitutes novelty. Indeed, different recommender systems use various interpretations of the term *novelty*. On the other hand, in most cases, it is considered to be a measurement of the number of new things that are recommended. Measuring the number of recommended items that originate from the long tail is a common approach. It is a sign that the system is capable of recommending to users things that they would have never been able to find on their own. As a result, the ability to accurately measure novelty is essential for a successful recommender [251].

Definition 9 (Entropy-Based Novelty)**.** *Let us define the top-k recommendation list again as* $\mathcal{L}_u^{(1,\dots,k)}$*. Entropy-Based Novelty EBN@k is a metric that measures the*

capability of the system to suggest long-tail items.

$$EBN_u@k = -\sum_{i \in \mathcal{L}_u^{(1,\dots,k)}} p_i \cdot \log_2 p_i \tag{2.13}$$

where

$$p_i = \frac{|\{u \in U \mid i \text{ is relevant for user } u\}|}{|U|} \tag{2.14}$$

A small value of $EBN_u@k$ reflects an high level of novelty in recommendation lists [32].

Another novelty metric is *Expected Popularity Complement* which corresponds to the expected number of relevant items that come from the long tail [251].

Definition 10 (Expected Popularity Complement)**.** *Given a discount function $disc(k)$, a relevance index for a specific item to the user defined as $p(rel|i_k, u)$ and $(1-p(seen|i_k))$ as factor of item novelty Expected Popularity Complement (EPC) is defined as:*

$$EPC = C \sum_{i_k \in \mathbb{R}} disc(k) p(rel|i_k, u)(1 - p(seen|i_k)) \tag{2.15}$$

Moreover, if we excluded *rank* and *relevance* from the formula 2.15 and considered a measure that is somehow related to the concept of self-information – coming from the field of information theory – we would have a new measure of novelty more closely related to the concept of catalogue exploration [296].

Definition 11 (Expected Free Discovery)**.** *Given a top-k recommendation list $\mathcal{L}_u^{(1,\dots,k)}$, and the log of the inverse discovery distribution $p(i|seen)$ it is possible to compute the Expected Free Discovery (EFD) (or expected inverse collection frequency) of relevant and seen items as follow:*

$$EFD = -\frac{1}{\left|\mathcal{L}_u^{(1,\dots,k)}\right|} \sum_{i \in \mathcal{L}_u^{(1,\dots,k)}} \log_2 p(i|seen) \tag{2.16}$$

Algorithmic Bias

The problem of unfair outputs in machine learning applications is well studied [46, 76] and it has been extended also to recommender systems [172]. Indeed, bias and fairness analysis is gaining momentum in the last years [71, 195], since it unveils several essential aspects of the recommenders' behavior.

One of the possible fairness problems with recommender systems, which has already come up in the previous sections, is connected to the *popularity bias* [34, 44].

Collaborative filtering recommenders, which will be used along this thesis, typically emphasize popular items and leave the long-tail items underrepresented. Unfortunately, delivering only popular items does not enhance new item discovery and may be unfair to the creators of less popular or newer items since a small set of users rates them.

Three bias metrics will be used in this dissertation to evaluate the degree of underrepresentation of items from the long-tail. The first metric, whose abbreviation is ACLT, measures the fraction of the long-tail items the recommender has covered.

Definition 12 (Average Percentage of Long Tail Items [6])**.** *Given the premises in Definition 7, it is defined as:*

$$ACLT@k = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} |\mathcal{L}_u^{(1,\ldots,k)} \cap \Gamma|, \tag{2.17}$$

where Γ *is the set containing the long-tail items.*

Alongside this metric, we can define two other metrics that analyze the exposure of items in the long tail: Average Percentage of Long Tail (APLT)[6] Items and Average Recommendation Popularity (ARP) [283]. The former is directly related to ACLT formulation and expresses the measures of the average popularity of long-tail items in the top-k recommendations of users, the latter calculates the popularity of items in a list of recommendations based on the number of interactions of each item in training.

Moreover, we have also evaluated PopREO and PopRSP, which are specific applications of RSP and REO [300]. PopREO estimates the equal opportunity of items, encouraging the true positive rate of popular and unpopular items to be the same. PopRSP is a measure of statistical parity, assessing whether the ranking probability distributions for popular and unpopular items are the same in a recommendation. For both measures, lower values indicate that the recommendations are less biased.

Another factor that may impact fairness is the *bias disparity*, which represents the degree to which a group's preferences on various item categories fail to be reflected in the recommendations they receive [47]. This disparity among users will degrade users' satisfaction, loyalty, and effectiveness of recommender system. In detail, in this dissertation, we are sometimes interested in checking whether the recommendation list deviate from the users' original preferences towards different item categories. In order to do that, we measure the bias disparity defined in Mansoury et al. [172] considering a unique group of users.

Definition 13 (Bias disparity)**.** *Let* $\{\mathcal{C}_1,\ldots,\mathcal{C}_P\}$ *a partition of* $\mathcal{I}$ *into* P *categories of items. Then, the bias disparity on a category of items* $\mathcal{C}_i$ *is defined as:*

$$BD(\mathcal{C}_i) = \frac{B_R(\mathcal{C}_i) - B_T(\mathcal{C}_i)}{B_T(\mathcal{C}_i)}, \tag{2.18}$$

where $B_T(\mathcal{C}_i)$ *is the source bias on category* $\mathcal{C}_i$*, i.e., how much users were biased towards category* $\mathcal{C}_i$ *in the training set, and* $B_R(\mathcal{C}_i)$ *is the bias on* $\mathcal{C}_i$ *in recommendation lists. In detail:*

$$B_T(\mathcal{C}_i) = \frac{\sum_{u\in\mathcal{U}} |\mathcal{C}_i \cap \mathcal{I}_u^{+,\text{train}}|}{\sum_{u\in\mathcal{U}} |\mathcal{I}_u^{+,\text{train}}|} \frac{1}{\frac{|\mathcal{C}_i|}{|\mathcal{I}|}}, \tag{2.19}$$

where the first term is the preference ratio of users on category $\mathcal{C}_i$*, and the denominator of the second term is the ratio of item category* $\mathcal{C}_i$ *in the dataset. Analogously:*

$$B_R(\mathcal{C}_i) = \frac{\sum_{u\in\mathcal{U}} |\mathcal{C}_i \cap \mathcal{L}_u|}{\sum_{u\in\mathcal{U}} |\mathcal{L}_u|} \frac{1}{\frac{|\mathcal{C}_i|}{|\mathcal{I}|}}. \tag{2.20}$$

Bias disparity shows positive or negative values if the recommender systems deviate the users respectively towards or away from a certain category. Thus, the closer to 0, the more the recommendation lists reflect the users' behavior encoded in the data.

Chapter 3

ELLIOT a rigorous and robust framework for Recommender Systems evaluation

It has been demonstrated that recommender systems are an efficient method to alleviate the problem of having too many options to choose from and to provide accurate and personalized recommendations. On the other hand, due to the extraordinary number of recommendation algorithms, splitting strategies, evaluation protocols, metrics, and tasks that have been proposed, rigorous experimental evaluation has proven to be an especially difficult endeavor. Because we were perplexed and frustrated by the constant need to recreate relevant evaluation benchmarks, experimental pipelines, hyperparameter optimization, and evaluation procedures, we decided to develop an exhaustive framework to meet such needs. In this section we present Elliot, a comprehensive recommendation framework that aims to run and reproduce an entire experimental pipeline by processing a simple configuration file. This goal can be accomplished through the use of the Elliot system. The data are loaded, filtered, and divided by the framework while taking into consideration a very large number of different strategies (13 splitting methods and 8 filtering approaches, from temporal training-test splitting to nested K-folds Cross-Validation). Elliot[1] optimizes the hyperparameters (51 strategies) for a number of different recommendation algorithms (more than 60 to date), selects the best models, compares them with the baselines providing intra-model statistics, computes metrics (36 possible choices) spanning from

[1]https://github.com/sisinflab/elliot

accuracy to beyond-accuracy, bias, and fairness, and conducts statistical analysis (Wilcoxon and Paired t-test).

3.1 Introduction

In the last decade, Recommendation Systems (RSs) have gained momentum as the pivotal choice for personalized decision-support systems. Recommendation is essentially a retrieval task where a catalog of items is ranked and the top-scoring items are presented to the user [143]. Once it was demonstrated their ability to provide personalized items to clients, both Academia and Industry devoted their attention to RSs [28, 37, 163, 164]. This collective effort resulted in an impressive number of recommendation algorithms, ranging from memory-based [221] to latent factor-based [66, 90, 140, 196], and deep learning-based methods [156, 270]. At the same time, the RS research community realized that focusing only on the accuracy of results could be detrimental and started exploring beyond-accuracy evaluation [251]. The community became conscious that *accuracy* was not sufficient to guarantee user satisfaction [177]. *Novelty* and *diversity* [51, 122, 250] came into play as new dimensions to be analyzed when comparing algorithms. However, this was only the first step in the direction of a more comprehensive evaluation of RSs. Indeed, more recently, the presence of *biased* [30, 298] and *unfair* [70, 79, 80] recommendations towards user groups and item categories has been widely investigated. This flourishing of metrics added several degrees of freedom to the recommendation evaluation. Models and evaluation aspects notwithstanding, a researcher has many other decisions to ponder. In fact, RSs have been widely studied and applied in various domains and tasks, with different (and often contradicting in their hypotheses) splitting preprocessing strategies [49] fitting the specific scenario. Moreover, machine learning (and recently also deep learning) techniques are prominent in algorithmic research and require their hyperparameter optimization strategies and procedures [25, 248].

The abundance of possible choices generated much confusion about choosing the correct baselines, conducting the hyperparameter optimization and the experimental evaluation [214, 215], and reporting the details of the adopted procedure. Consequently, two major concerns arose: unreproducible evaluation and unfair comparisons [236]. On the one hand, the negative effect of unfair comparisons is that various proposed recommendation models have been compared with suboptimal baselines [69, 205]. On the other hand, in a recent study [69], it has been shown that only one-third of the published experimental results are, in fact, reproducible. Moreover, providing trained models, recommendation lists, and results to reviewers is impractical. The

problem of reproducing the experiments recurs when the need to recompute the whole set of experiments (e.g., the addition of a model) emerges. Whether or not it is a new experiment, it opens the doors to another class of problems: the number of possible design choices often imposes the researcher to define and implement only the chosen setting. As a result, any variation imposes the implementation of other models, experimental settings, and metrics.

Progressively, the RS community has welcomed the emergence of recommendation, evaluation, and even hyperparameter tuning frameworks [41, 78, 94, 236, 250]. However, facilitating reproducibility or extending the provided functionality would typically depend on developing bash scripts or programming on whatever language each framework is written. This, together with other issues mentioned before (e.g., too specific experimental settings, small sets of algorithms or metrics) make most of these frameworks not flexible enough in a fast moving environment such as RS research.

This work introduces Elliot, a novel kind of recommendation framework, to overcome these obstacles. The framework analyzes the recommendation problem from the researcher's perspective. Indeed, Elliot conducts a whole experiment, from dataset loading to results gathering. The core idea is to feed the system with a simple and straightforward configuration file that drives the framework through the experimental setting choices. Elliot untangles the complexity of combining splitting strategies, hyperparameter model optimization, model training, and the generation of reports of the experimental results. The rationale is to keep the entire experiment reproducible and put the user (in our case, a researcher or RS developer) in control of the framework. Elliot natively provides for widespread research evaluation features, like the analysis of multiple cut-offs and several RSs (50). According to the recommendation model, the framework allows, to date, the choice among 27 similarities, the definition of multiple neural architectures, and 51 hyperparameter tuning combined approaches, unleashing the full potential of the HyperOpt library [41]. To enable the evaluation for the diverse tasks and domains, Elliot supplies 36 metrics (including Accuracy, Error-based, Coverage, Novelty, Diversity, Bias, and Fairness metrics), 13 splitting strategies, and 8 prefiltering policies. The framework can also measure to what extent the RS results are significantly different from each other, providing the paired t-test and Wilcoxon statistical hypothesis tests. Finally, Elliot lets the researcher quickly build their models and include them in the experiment.

3.2 Related Works and Frameworks

RS evaluation is an active, ever-growing research topic related to reproducibility, which is a cornerstone of the scientific process as identified by Konstan et al. [137]. Recently researchers have taken a closer look at this problem, in particular because depending on how well we evaluate and assess the efficacy of a system, the significance and impact of such results will increase.

Some researchers argue that to enhance reproducibility, and to facilitate fair comparisons between different works (either frameworks, research papers, or published artifacts), at least the following four stages must be identified within the evaluation protocol [214]: *data splitting*, *item recommendations*, *candidate item generation*, and *performance measurement*. In a recent work [35], these stages have been completed with *dataset collection* and *statistical testing*. Some of these stages can be further categorized, such as performance measurement, depending on the performance dimension to be analyzed (e.g., ranking vs error, accuracy vs diversity, and so on).

In fact, the importance and relevance of the aforementioned stages have been validated in recent works; however, even though most of the RS literature has been focused on the impact of the item recommendation stage as an isolated component, this is far from being the only driver that affects RS performance or the only component impacting on its potential for reproducibility. In particular, Meng et al. [179] survey recent works in the area and conclude that no standard splitting strategy exists, in terms of random vs temporal splits; furthermore, the authors found that the selection of the splitting strategy can have a strong impact on the results. Previously, Campos et al. [49] categorized and experimented with several variations of random and temporal splitting strategies, evidencing the same inconsistency in the results. Regarding the candidate item generation, it was first shown [33] that different strategies selecting the candidate items to be ranked by the recommendation algorithm may produce results that are orders of magnitude away from each other; this was later confirmed [214] in the context of benchmarking recommendation frameworks. Recent works [143, 154] evidenced that some of these strategies selecting the candidate items may introduce inconsistent measurements which should, hence, not be trusted.

Finally, depending on the recommendation task and main goal of the RS, several performance dimensions, sometimes contradicting, can be assessed. For a classical overview of these dimensions, we refer the reader to Gunawardana et al. [98], where metrics accounting for prediction accuracy, coverage, confidence, trust, novelty, diversity, serendipity, and so on are defined and compared. However, to the best of our knowledge, there is no public implementation providing more than one or two of these dimensions.

Moreover, recently the community has considered additional dimensions such as bias (in particular, popularity bias [1]) and fairness [80]. These dimensions are gaining attention, and several metrics addressing different subtleties are being proposed, but no clear winner or standard definition emerged so far – as a consequence, the community lacks an established implementation of these novel evaluation dimensions.

Reproducibility is the keystone of modern RSs research. Dacrema et al. [69] and Rendle et al. [201] have recently raised the need of comprehensive and fair recommender model evaluation. Their argument on the outperforming recommendation accuracy of latent-factor models over deep-neural ones, when an extensive hyper-parameter tuning was performed, made it essential the development of novel recommendation frameworks. Starting from 2011, Mymedialite [94], LensKit [78, 82], LightFM [145], RankSys [250], and Surprise [121], have formed the basic software for rapid prototyping and testing of recommendation models, thanks to an easy-to-use model execution and the implementation of standard accuracy, and beyond-accuracy, evaluation measures and splitting techniques. However, the outstanding success and the community interests in deep learning (DL) recommendation models, raised need for novel instruments. LibRec [99], Spotlight [146], and OpenRec [277] are the first open-source projects that made DL-based recommenders available – with less than a dozen of available models without filtering, splitting, and hyper-optimization tuning strategies. An important step towards more exhaustive and up-to-date set of model implementations have been released with RecQ [281], DeepRec [101], and Cornac [217] frameworks. However, they do not provide a general tool for extensive experiments on the pre-elaboration and the evaluation of a dataset. Indeed, after the reproducibility hype [69, 201], DaisyRec [236] and RecBole [292] raised the bar of framework capabilities, making available both large set of models, data filtering/splitting operations and, above all, hyper-parameter tuning features. However, we found a significant gap in splitting and filtering capabilities, in addition to a complete lack of two nowadays popular (even critical) aspects of recommendation performance: biases and fairness. Reviewing these related frameworks, emerged a striking lack of an open-source recommendation framework able to perform *by design* an extensive set of pre-elaboration operations, to support several hyperparameters optimization strategies and multiple sets of evaluation measures, which include bias and fairness ones, supported by statistical significance tests – a feature absent in other frameworks (as of February 2021). Elliot meets all these needs. Table 3.1, table 3.2, and table 3.3 give an overview of the frameworks and to which extent they satisfy the mentioned requirements [2].

[2]Statistics available in these tables are updated to the beginning of March 2022

Table 3.1 Overview of the ELLIOT and related frameworks functionalities for data elaboration and model optimization stragies.

DATA ELABORATION AND MODEL OPTIMIZATION STRATEGIES																																					
	Prefiltering									**Splitting**																					**Hyperparam. Tuning**								
	Filter-by-rating			*k*-core						Temporal							Random										Fix		Search Strategy				Search Space					
										Fix		*HO*					*HO*					*K-HO*				*CV*												
	Numerical	Distributional	User Dist.	User	Item	Iterative	Iter-*n*-rounds	Cold-Users	**Tot.**	Best timestamp	Handcrafted	By-Ratio Sys.	By-Ratio User	Leave-1-out	Leave-*n*-in	Leave-*n*-out	By-Ratio Sys.	By-Ratio User	Leave-1-out	Leave-*n*-in	Leave-*n*-out	By-Ratio Sys.	Leave-1-out	Leave-*n*-out	*k*-folds Sys.	*k*-folds User	Computed	**Tot.**	Grid	Annealing	Bayesian	Random	Fix	Uniform	Normal	Log-Uniform	**Tot.**
LensKit Java [82]								0				✓	✓	✓	✓	✓	✓	✓	✓	✓	✓					✓	✓	10									0
Mymedialite [94]								0			✓	✓			✓	✓								✓		✓	6									0
RankSys [250]								0																		✓	1									0
LibRec [99]								0			✓	✓	✓	✓		✓	✓	✓	✓			✓		✓		✓	11									0
Implicit [87]								0																		✓	1									0
OpenRec [277]								0																		✓	1									0
RecQ [281]								0										✓				✓				✓	3									0
DeepRec [101]								0									✓									✓	2									0
LensKit Python [78]								0															✓	✓	✓	✓	✓	5									0
Surprise [121]								0										✓	✓			✓	✓		✓		✓	6	✓			✓	✓					3
Cornac [217]	✓			✓	✓			3									✓	✓							✓		✓	4	✓			✓	✓	✓				4
RecBole [292]								0				✓	✓			✓	✓	✓	✓		✓					✓	✓	8	✓	✓	✓	✓	✓	✓		✓		7
DaisyRec [236]	✓			✓	✓			3			✓	✓	✓			✓	✓	✓								✓	7			✓		✓	✓		✓		4
ELLIOT	✓	✓	✓	✓	✓	✓	✓	✓	**8**	✓	✓		✓	✓		✓		✓	✓		✓	✓	✓	✓		✓	✓	**13**	✓	✓	✓	✓	✓	✓	✓	✓	**8**

Table 3.2 Overview of the Elliot and related frameworks available models.

RECOMMENDER MODELS																																								
	Popular Models																																							
	MostPop	Random	I-kNN[159]	U-kNN[206]	PureSVD[140]	FunkSVD[90]	SVD++[139]	SLIM[184]	MF[141]	WRMF[287]	FM[196]	NeuMF[114]	BPRMF[197]	DMF[276]	FISM[132]	NNMF[166]	NFM[108]	SoREC[167]	DeepFM[100]	PMF[218]	AFM[273]	FFM[131]	WBPR[93]	DSSM[120]	VBPR[107]	ConvMF[135]	GMF[114]	Caser[238]	MultiVAE[156]	I-AutoR[224]	U-AutoR[224]	CDAE[270]	CML[118]	LogMF[130]	LDA[142]	SoMF[124]	SlopeOne[151]	SoReg[168]	**Others**	**Tot.**
LensKit Java [82]			✓	✓		✓			✓																												✓		0	5
Mymedialite [94]	✓	✓	✓	✓			✓	✓	✓	✓			✓																					✓		✓	✓		6	18
RankSys [250]	✓	✓	✓	✓					✓		✓									✓															✓				0	8
LibRec [99]	✓	✓	✓	✓			✓	✓	✓	✓	✓		✓		✓	✓	✓	✓		✓		✓	✓			✓				✓	✓	✓			✓	✓	✓	✓	30	55
Implicit [87]			✓	✓									✓																					✓					0	4
OpenRec [277]													✓							✓					✓								✓						7	11
RecQ [281]					✓		✓											✓		✓																✓	✓	✓	6	13
DeepRec [101]									✓		✓	✓	✓	✓		✓	✓				✓			✓			✓	✓		✓	✓	✓	✓						5	20
LensKit Python [78]	✓		✓	✓		✓			✓				✓																										0	6
Surprise [121]			✓	✓	✓		✓									✓																					✓		1	7
Cornac [217]	✓		✓	✓	✓				✓			✓				✓		✓		✓			✓		✓	✓	✓		✓										21	35
RecBole [292]	✓		✓								✓	✓	✓	✓	✓		✓		✓		✓	✓		✓				✓	✓			✓							50	**65**
DaisyRec [236]	✓		✓	✓	✓			✓	✓	✓	✓	✓					✓		✓		✓								✓			✓							0	14
Elliot	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓			✓	✓	✓		✓	✓	✓		✓	✓			✓		20	50

Table 3.3 Overview of the ELLIOT and related frameworks functionalities for evaluation step.

EVALUATION OF RECOMMENDATION PERFORMANCE											
	Metric Families								Stat. Tests		
	Accuracy	Error	Coverage	Novelty	Diversity	Bias	Fairness	**Tot.**	Paired t-test	Wilcoxon	**Tot.**
LensKit Java [82]	2	2	1		1			6			0
Mymedialite [94]	6	2						8			0
RankSys [250]	7		**3**	**6**	**18**			34			0
LibRec [99]	6	**4**						10			0
Implicit [87]								0			0
OpenRec [277]	4							4			0
RecQ [281]	6	2						8			0
DeepRec [101]	6	2						8			0
LensKit Python [78]	4	2						6			0
Surprise [121]		**4**						4			0
Cornac [217]	8	3						11			0
RecBole [292]	8	3						11			0
DaisyRec [236]	8							8			0
ELLIOT	**11**	3	**3**	2	3	**10**	**4**	**36**	✓	✓	**2**

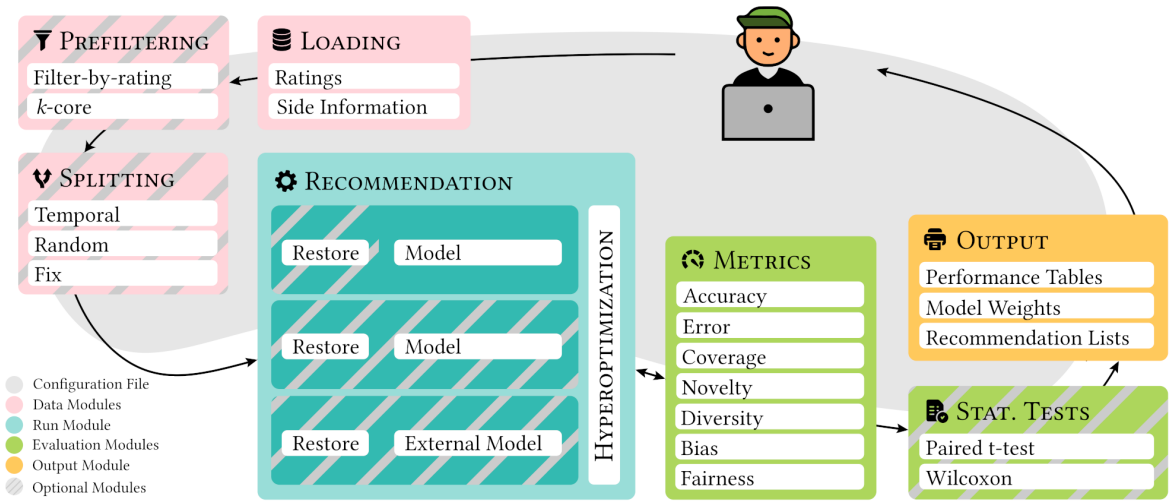


Fig. 3.1 Overview of ELLIOT.

3.3 Framework details

ELLIOT is an extensible framework composed of eight functional modules, each of them responsible for a specific step of an experimental recommendation process. What

happens under the hood (Figure 3.1) is transparent to the user, who is only expected to provide human-level experimental flow details using a customizable configuration file. Accordingly, ELLIOT builds the overall pipeline. The following sections deepen into the details of the eight ELLIOT's modules and outline the preparation of a configuration file.

3.3.1 Data Preprocessing

The data module is devoted to handle and manage the input of the experiment, e.g., the user-item rating matrix. ELLIOT supports two types of data inputs: standard user-item preference scores, e.g., ratings and users, and items side information, e.g., movie genres. After the loading stage, the input data go through two modules: filtering and splitting. For instance, the former can drop offcold users— users with less than k-feedback— the latter can split the dataset in training, validation, and test by specifying the splitting ratios, e.g., 8:1:1. Next, we provide mode details on the ELLIOT loading, filtering, and splitting modules.

Loading. As stated before, RSs experiments could require multiple data sources such as user-item feedback, e.g., ratings, or additional side information, e.g., the visual features of items' images. The framework provides a dedicated module to manage the data loading to fulfill these requirements, referred to as `DataLoader` (`DL`). DL loads the standard user-item interaction files as wither a single file to be filtered and split later or already pre-elaborated ones, e.g., the system user might pre-filter and split the data with custom procedures. Users and items additional data handling is performed by data-driven extension of DL. For instance, the `VisualDataLoader` performs the loading of item visual features, e.g., extracted by a convolutional neural network, to build visually-aware recommender models [57, 134]. Besides, the `KnwoladgeDataLoader` manages semantic features extracted from knowledge graphs to be integrated into knowledge-aware recommenders [26]. Note that, in the case of recommender models that utilize side information, the `DataLoader` will filter out all the users and items without it, e.g., an item will be filtered out if it is missing its visual feature in the case of experiments involving visual recommenders.

Prefiltering. After the loading of data, ELLIOT provides data-filter operations. We will refer to this framework feature as `DataFilter (DF)`. `DF` makes available two filtering strategies: filter-by rating threshold (`Rating Thld.`) and k-`core` filtering. `Rating Thld.` drops offa user-item interaction if the rating/preference score is smaller than the threshold (t). It is possible to set t specifying (i) a `Numerical` value, e.g., 3.5, (ii) a `Distributional` detail, e.g., global rating average value, and (iii) a user-

based distributional (`Dist-User`) values, e.g., user's average rating value. The k-`core` strategy filters out the users and/or items will less than k recorded interactions. It can be specified whether the filtering is employed on either or both of `User` and `Item` by optionally specifying to iterate (`Iterative`) until the k-`core` filtering is achieved, e.g., all the users and items have at least k recorded interaction. Since the reaching of the k-`core` conditions might be intractable, there are not cases where users and items have at least k recorded, ELLIOT allows to specify the maximum number of iterations (`Iter-`n`-rounds`), e.g., $n = 10$. Finally, the system users could also specify to filter out, or not, the cold-users by choosing the `Cold-Users` filtering feature.

Splitting. If needed, the data is served to the *Splitting* module. In detail, ELLIOT provides (i) *Temporal*, (ii) *Random*, and (iii) *Fix* strategies. The *Temporal* strategy splits the user-item interactions based on the transaction timestamp, i.e., fixing the timestamp, finding the optimal one [27, 36], or adopting a hold-out (*HO*) mechanism. The *Random* strategy includes hold-out (*HO*), K-repeated hold-out (K-*HO*), and cross-validation (*CV*). Table 3.1 provides further configuration details. Finally, the *Fix* strategy exploits a precomputed splitting.

3.3.2 Recommendation Models

fter data loading and pre-elaborations, *Recommendation* module (Figure 3.1) provides the functionalities to train (and restore) the ELLIOT recommendation models and the new ones integrated by users.

Implemented Models. ELLIOT integrates, to date, more than 50 recommendation models (see Table 3.2) partitioned into two sets. The first set includes 38 *popular* models implemented in at least two of frameworks reviewed in this work (i.e., adopting a framework-wise popularity notion). Table 3.2 shows that ELLIOT is the framework covering the largest number of popular models, with 30 models out of 38, i.e., 79%. The second set comprises other well-known state-of-the-art recommendation models implemented in less than two frameworks, namely, BPRSLIM [184], ConvNCF [110], NPR [185], MultiDAE [156], and NAIS [112], graph-learning based, i.e., NGCF [261], and LightGCN [109], visual-based, i.e., VBPR [107], DeepStyle [161], DVBPR [134], ACF [57], and VNPR [185], adversarial-robust, i.e., APR [111] and AMR [239], generative adversarial network (GAN)-based, i.e., IRGAN [257] and CFGAN [52], content-aware, i.e., Attribute-I-kNN and -U-kNN [94], VSM [23, 187], Wide & Deep [60], and KaHFM [26] recommenders.

Hyper-parameter Tuning. Hyperparameter tuning is an ingredient of the recommendation model training that definitely influences its performance [201]. ELLIOT

provides *Grid Search*, *Simulated Annealing*, *Bayesian Optimization*, and *Random Search* strategies. Furthermore, Elliot allows performing four traversing strategies across the search space defined in each recommendation model configuration. When the user details the possible hyperparameters (as a list) without specifying a search strategy, Elliot automatically performs an exhaustive *Grid Search*. Besides the *Grid Search*, Elliot exploits the full potential of the *HyperOpt* [41] library, considering all its sampling strategies. Table 3.1 summarizes the available *Search Strategies* and *Search Spaces*.

3.3.3 Performance Evaluation

After the training phase, Elliot continues its operations, evaluating recommendations. Figure 3.1 indicates this phase with two distinct evaluation modules: Metrics and Statistical Tests.

Metrics. As shown in Table 3.3, 36 evaluation metrics are available in Elliot partitioned into seven families: *Accuracy* [223, 294], *Error*, *Coverage*, *Novelty* [251], *Diversity* [282], *Bias* [5, 6, 245, 279, 300], and *Fairness* [70, 299]. It is worth mentioning that Elliot is the framework that exposes both the largest number of metrics and the only one considering bias and fairness measures. Moreover, the user can choose any metric to drive the model selection and the tuning. Furthermore, it can be seen that Elliot is the model that exposes both the largest number o metrics and the only one considering the biases and fairness measures. Here, it is fundamental to mention that it is possible to specify one of these measures as the validation metric used to find the best parameters configuration during the hyper-optimization procedure. The best configuration is used by Elliot as the model from which extracts the recommendation lists and eventually performs the statistical test detailed below.

Statistical Tests As shown in Figure 3.1, Statistical Tests is the last optional module of Elliot before the `Output` module described in the following section. Table 3.3 shows that the reviewed related frameworks completely miss any statistical test procedure to support the experimental results statistically. In detail, Elliot brings the opportunity to compute two statistical tests, i.e., `Paired t-Test` and `Wilcoxon`. The test is performed between each pair of models, or best-tuned models, in the recommender models' set defined by the user using Elliot. Particularly, the Statistical Tests module activation involves only the activation of a flag in the configuration file.

3.3.4 Framework Outcomes

The last ELLIOT module deals with storing the output of the whole configured experiment. ELLIOT comes with a set of preconfigured output files which we expect to meet all users' requirements. In the following, we briefly survey the details of the files stored by ELLIOT.

ELLIOT gives the possibility to store three classes of output reports: (i) *Performance Tables*, (ii) *Model Weights*, and (iii) *Recommendation Lists*. The former consist of spreadsheets (in a *tab-separated-value* format) with all the metric values computed on the test set for every recommendation model specified in the configuration file. The tables comprise cut-offspecific and model-specific tables (i.e., considering each combination of the explored parameters). The user can also choose to store tables with the triple format, i.e., *<Model, Metric, Value>*. Tables also include cut-off-specific statistical hypothesis tests and a JSON file that summarizes the best model parameters. Optionally, ELLIOT saves model weights to avoid future re-training of the recommender. Finally, ELLIOT stores the top-k recommendation lists for each model adopting a tab-separated *<User, Item, Predicted Score>* triple-based format.

3.3.5 Preparation of the Experiment

The operation of ELLIOT is triggered by a single configuration file written in YAML. Configuration 3.1 shows a toy example of a configuration file. The file sections reflect module colors in Figure 3.1 to ease the understanding. The first section details the data loading, filtering, and splitting information as defined in Section 3.3.1. The `models` section represents the recommendation models configuration, e.g., Item-kNN, described in Section 3.3.2. Here, the model-specific hyperparameter optimization strategies are specified, e.g., the grid-search in Configuration 3.1. The `evaluation` section details the evaluation strategy with the desired metrics, e.g., nDCG in the toy example. Finally, `save_recs` and `top_k` keys detail, for example, the *Output* module abilities described in Section 3.3.4. It is worth noticing that, to the best of our knowledge, ELLIOT is the only framework able to run an extensive set of reproducible experiments by merely preparing a single configuration file. The next section exemplifies two real experimental scenarios commenting on the salient parts of the configuration files.

Listing 3.1 `hello_world.yml`

```yaml
experiment:
  dataset: movielens_1m
  data_config:
    strategy: dataset
    dataset_path: ../data/movielens_1m/dataset.tsv
  splitting:
    test_splitting:
        strategy: random_subsampling
        test_ratio: 0.2
  models:
    ItemKNN:
      meta:
        hyper_opt_alg: grid
        save_recs: True
      neighbors: [50, 100]
      similarity: cosine
  evaluation:
    simple_metrics: [nDCG]
  top_k: 10
```

3.4 Experimental Scenarios

This section illustrates how to prepare, execute and evaluate two comprehensive experimental scenarios with ELLIOT for a *basic* and a more *complex* experimental setting.

3.4.1 Basic Configuration

Experiment. In the first scenario, the experiments require comparing a group of RSs whose parameters are optimized via a grid-search. Configuration 3.2 specifies the data loading information, i.e., semantic features source files, in addition to the filtering and splitting strategies. In particular, the latter supplies an entirely automated way of preprocessing the dataset, which is often a time-consuming and non-easily-reproducible phase. The `simple_metrics` field allows computing accuracy and beyond-accuracy metrics, with two top-k cut-offvalues (5 and 10) by merely inserting the list of desired measures, e.g., `[Precision, nDCG, ...]`. The knowledge-aware recommendation model, `AttributeItemKNN`, is compared against two baselines: `Random` and `ItemKNN`, along with a user-implemented model that is `external.MostPop`. The configuration makes use of ELLIOT's feature of conducting a grid search-based hyperparameter optimization strategy by merely passing a list of possible hyperparameter values,

Listing 3.2 `basic_configuration.yml`

```
experiment:
    dataset: cat_dbpedia_movielens_1m
    data_config:
        strategy: dataset
        dataloader: KnowledgeChainsLoader
        dataset_path: <...>/dataset.tsv
        side_information:
            <...>
    prefiltering:
        strategy: user_average
    splitting:
        test_splitting:
            strategy: temporal_hold_out
            test_ratio: 0.2
        <...>
    external_models_path: ../external/models/__init__.py
    models:
        Random:
            <...>
        external.MostPop:
            <...>
        AttributeItemKNN:
            neighbors: [50, 70, 100]
            similarity: [braycurtis, manhattan]
            <...>
    evaluation:
        cutoffs: [10, 5]
        evaluation: [nDCG, Precision, ItemCoverage, EPC, Gini]
        relevance_threshold: 1
    top_k: 50
```

github.com/sisinflab/elliot/blob/master/config_files/basic_configuration.yml

e.g., `neighbors: [50, 70, 100]`. The reported models are selected according to *nDCG@10*.

Results. Table 3.4 displays a portion of experimental results generated by feeding ELLIOT with the configuration file. The table reports four metric values computed on recommendation lists at cutoffs 5 and 10 generated by the models selected after the hyperparameter tuning phase. For instance, Attribute-I-kNN model reports values for the configuration with `neighbors` set to `100` and `similarity` set to `braycurtis`. Table 3.4 confirms some common findings: the item coverage value ($ICov$@10) of an Attribute-I-kNN model is higher than the one measured on I-kNN, and I-kNN is the most accurate model.

Table 3.4 Experimental results for Configuration 3.2.

Model	*nDCG*@5	*ICov*@5	*nDCG*@10	*ICov*@10
Random	0.0098	3197	0.0056	3197
MostPop	0.0699	68	0.0728	96
I-*k*NN	0.0791	448	0.0837	710
Attribute-I-*k*NN	0.0464	1575	0.0485	2102

3.4.2 Advanced Configuration

Experiment. The second scenario depicts a more complex experimental setting. In Configuration 3.3, the user specifies an elaborate data splitting strategy, i.e., `random_subsampling` (for test splitting) and `random_cross_validation` (for model selection), by setting few splitting configuration fields. Configuration 3.3 does not provide a cut-offvalue, and thus a top- k field value of 50 is assumed as the cut-off. Moreover, the evaluation section includes the `UserMADrating` metric. ELLIOT considers it as a complex metric since it requires additional arguments (as shown in Configuration 3.3). The user also wants to implement a more advanced hyperparameter tuning optimization. For instance, regarding NeuMF, Bayesian optimization using *Tree of Parzen Estimators* [39] is required (i.e., `hyper_opt_alg: tpe`) with a logarithmic uniform sampling for the learning rate search space. Moreover, ELLIOT allows considering complex neural architecture search spaces by inserting lists of tuples. For instance, `(32, 16, 8)` indicates that the neural network consists of three hidden layers with 32, 16, and 8 units, respectively.

Results. Table 3.5 provides a summary of the experimental results obtained feeding ELLIOT with Configuration 3.3. Even here, the columns report the values for all the considered metrics (simple and complex metrics). Configuration 3.3 also requires statistical hypothesis tests. Therefore, the table reports the *Wilcoxon-test* outcome (computed on pairs of models with their best configuration).

Table 3.5 Experimental results for Configuration 3.3.

Model	*nDCG*@50	*ARP*@50	*ACLT*@50	$UMAD_H$@50
BPRMF	0.2390	1096	0.0420	0.0516
NeuMF	0.2585	919	0.8616	0.0032
MultiVAE	0.2922†	755†	3.2871†	0.1588

†*p-value* ≤ 0.001 *using Wilcoxon-test*

3.5 Summary

Elliot is a framework that examines the recommendation process from an RS researcher's perspective. It requires the user just to compile a flexible configuration file to conduct a rigorous and reproducible experimental evaluation. The framework provides several loading, prefiltering, splitting, hyperparameter optimization strategies, recommendation models, and statistical hypothesis tests. Elliot reports can be directly analyzed and inserted into research papers. We reviewed the RS evaluation literature, positioning Elliot among the existing frameworks, and highlighting its advantages and limitations. Next, we explored the framework architecture and how to build a working (and reproducible) experimental benchmark. To the best of our knowledge, Elliot is the first recommendation framework that provides a full multi-recommender experimental pipeline based on a simple configuration file. We plan to extend soon Elliot in various directions to include: sequential recommendation scenarios, adversarial attacks, reinforcement learning-based recommendation systems, differential privacy facilities, sampled evaluation, and distributed recommendation.

Listing 3.3 `advanced_configuration.yml`

```
experiment:
    dataset: movielens_1m
    data_config:
        strategy: dataset
        dataset_path: <...>/dataset.tsv
    prefiltering:
        strategy: iterative_k_core
        core: 10
    splitting:
        test_splitting:
            strategy: random_subsampling
            test_ratio: 0.2
        validation_splitting:
            strategy: random_cross_validation
            folds: 5
    models:
        BPRMF:
            <...>
        NeuMF:
            meta:
                hyper_max_evals: 5
                hyper_opt_alg: tpe
            lr: [loguniform, -10, -1]
            mf_factors: [quniform, 8, 32, 1]
            mlp_hidden_size: [(32, 16, 8), (64, 32, 16)]
            <...>
        MultiVAE:
            <...>
    evaluation:
        simple_metrics: [nDCG, ARP, ACLT]
        wilcoxon_test: True
        complex_metrics:
        - metric: UserMADrating
            clustering_name: Happiness
            clustering_file: <...>/u_happy.tsv
        relevance_threshold: 1
    top_k: 50
```

github.com/sisinflab/elliot/blob/master/config_files/advanced_configuration.yml

Chapter 4

Reproducibility analysis on State-Of-The-Art Collaborative Filtering models

Research on RSs algorithms, much like research in other fields of applied machine learning, is primarily driven by attempts to enhance the state-of-the-art, most often in terms of accuracy measurements. However, some recent research studies reveal that the purported advances that have been made over the years frequently "don't add up", and that approaches that were published several years ago often perform better than the most current models when they are independently reviewed. Several variables, including the fact that some researchers presumably just fine-tune their own models and do not adjust the baselines, contribute to this issue.

The contribution of this chapter is twofold: on the one hand, we will show how it is possible to correctly replicate an experiment with the Elliot framework when the set-up is clear and well-defined. On the other hand, we will carefully study the performance of the major RSs, belonging to the CF family, which currently represent the state of the art for this research area. In addition, we will integrate an analysis on novelty/diversity and bias/fairness measures in both of these study phases. Through this investigation we will verify how often there is no clear predominance of neural network-based models. Furthermore, we can confirm that there is no absolute ranking of recommendation algorithms for all case histories, but the ranking varies according to the metrics and dataset under investigation. However, what we can see is how traditional matrix factorization, linear models, and nearest neighbor-based models often dominate the leaderboard across datasets and metrics.

4.1 Introduction

In today's world, recommender systems are becoming more common in online applications. These systems are designed to assist users in discovering valuable information when approached with an overwhelming amount of data. Research in this area is thriving as a result of the high practical significance of such systems, notably in the fundamental machine learning (ML) techniques required to develop individualized item recommendations. Offline experimentation, in which the predictive or ranking accuracy of several machine learning models is evaluated, has been the technique of choice in recent years. The common objective of these types of research projects is to advance the "state of the art," and proof is then offered by reporting improvements over previously established models that were acquired in those studies.

Unfortunately, a number of recent research works published in the field of recommender systems and other related areas of applied ML research, such as information retrieval, indicate that some of these improvements that have been reported over the years "don't add up" [29]. Ferrari Dacrema et al. [85] for example, compares a number of recently developed "top-n" recommendation models against models developed in the past, which are often more straightforward. Since a result of their research, they discovered that a significant portion of the improvement that has been claimed seems to be simply "virtual," as the most recent models are nearly always surpassed by approaches that have already been developed (for a similar study, see also Rendle et al. [199]). This puzzling occurrence may have been caused by a number of factors, such as the selection of baselines that are (insufficiently) weak ([150, 157]) or the absence of baselines that have not been tuned appropriately. In addition, it often turns out that there is no obvious winner across datasets and accuracy metrics when such assessments are conducted independently, which means that they are not carried out by the authors of the techniques that are being compared. Given that the ranking of algorithms appears to depend on the particular experimental configuration in terms of baselines, accuracy measures, or datasets, it is still unclear what exactly represents the actual state-of-the-art in this field. Consequently, it is still unclear what constitutes the actual state-of-the-art in this field.

Our goal is to provide insights regarding what represents the state-of-the-art for *"top-n"* recommendation tasks, at least for those experimental settings that are common in the recent literature. Specifically, our focus will be on evaluating the performance of a number of different methods. Similar to what was done in Ferrari Dacrema et al. [85], we take into account a wide variety of collaborative filtering algorithms. These algorithms include both more traditional approaches based on nearest-neighbors,

various approaches to matrix factorization, linear models, as well as more contemporary methods based on deep learning. In contrast to earlier evaluations such as Ferrari Dacrema et al. [85], however, we benchmarked all of the algorithms under the exact same experimental conditions, that is, with the same datasets and utilizing the exact same evaluation protocol. This was done after methodically adjusting the hyperparameters of each model to achieve their highest level of efficiency.

The results of our studies indicate that none of the two latest neural approaches was the algorithm with the greatest performance in any of the scenarios that were taken into consideration. In addition, the ranking of the algorithms differs among datasets and assessment metrics, which is to be anticipated based on the research that has been done. We were caught offguard when we saw that linear models, closest neighbors, and classical matrix factorization are at the top of the leaderboard for all of the datasets and performance measures that we looked at. As a result of this, one of the takeaways from our study is that the non-neural approaches that ranked highest in our analysis should be taken into consideration as baselines in any future research conducted on recommendation algorithms.

It is important to point out that the datasets that were utilized in our studies were selected on the basis of the standard procedure that is currently being followed in the academic literature. However, in our opinion, these datasets are on the smaller side, and it is possible that bigger datasets will provide a different set of findings. Our current study, which intends to give insights on the state-of-the-art in regularly used assessment settings, does not concentrate on doing an analysis of this kind, however; rather, its primary objective is to do so. However, with this study, we present a collection of models that are fine-tuned for these typical datasets. As a result, the amount of effort that other researchers need to put in to adjust these baselines for their own studies is reduced. Our comprehension of the current state of the art in this field will be continually expanded as a result of our plans to produce fine-tuned models in the future that are also applicable to bigger datasets.

The starting point of this contribution is comparing Neural Networks (NNs) techniques against classical personalized algorithms — mostly matrix factorization or nearest neighbors —, emphasizing the lack of well-tuned baselines or incorrect, incomplete, or even unfair experimental conditions evidenced in the literature. Nonetheless, while these conclusions are useful to move forward on understanding when NNs should be applied in recommendation, they neglect evaluation dimensions that are important in the RS community, such as diversity, novelty, coverage, and so on [98], since most of

the authors have focused, so far, on the precision/accuracy of the recommended items produced by those methods.

In this context, we aim to bridge this gap and compare NNs against classical RSs under several evaluation dimensions. With this goal in mind, we focus on a recent paper [200] where the authors showed how proper hyperparameter selection could make simple operations like a dot product outperform similarity learning through NNs. In this respect, we have the following two main goals: first, **replicating** the aforementioned paper, since it has a salient characteristic where the authors used in their tables results from other papers (claiming they used comparable evaluation settings and tuning, something that too often is not true as it is difficult to do properly [214]); once we are able to replicate these results, we **reproduce** them under different situations. In particular, we report beyond-accuracy evaluation metrics, to explore the extent these methods behave on complementary dimensions they have not been optimised for, or whose results have not been reported.

4.2 Replication of prior experiments: settings and results

This section focuses on describing how the replication of the experiments from papers Dacrema et al. [68], He et al. [113], and Rendle et al. [200] has been set up. It starts by defining the background of the recommendation problem and later reviews Matrix Factorization and Neural Collaborative Filtering approaches. Following this first part, the evaluation protocol applied to compare Neural Collaborative Filtering (NCF) and Matrix Factorization (MF) against the baselines in their respective works is contextualized.

4.2.1 Background and Formulation

The notation used herein is summarised as follows. Matrices are denoted by uppercase letters A, vectors by lowercase bold letters $\mathbf{b}$, scalars by lowercase letters a. We denote the concatenation of the vectors $\mathbf{b}$ and $\mathbf{c}$ by $[\mathbf{b}, \mathbf{c}]$. Let there be a pool of users (U) to recommend to and a catalog of items (I) to recommend from. A recommendation algorithm returns a *score* for a given user-item pair that corresponds to the estimated degree of *satisfaction* for the user enjoying that item. In this work we focus on a specific kind of recommendation algorithms, where two d-dimensional embedding vectors, p and q, are combined into a single score. Conventionally, p represents the embedding of

a user, q the embedding of an item, and $\phi(p, q)$ ($\phi : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$) the similarity of the user to the item.

Matrix Factorization (MF) is a famous and classical example of model-based Collaborative Filtering methods [140]. The algorithm learns a latent representation of items and users, whose linear interactions aim to explain the observed feedback. There are several variations of MF proposed in the literature, and a comprehensive review would deserve a specific study that is out of the scope of this work. However, to provide the reader an intuition of how much the factorization strategy has been disruptive in recent years, we briefly review the works that are, in our humble opinion, the most representative or the ones that can show the myriad of possible applications of factorization models.

The first examples of factorization models were soon recognized as state-of-the-art models. Among these pioneering works, there could be found SVD [140], PureSVD [66], SVD++ [139], PMF [218, 219], NNMF [166], and SLIM [184]. Among the several methods on matrix factorization, Rendle's work has heavily influenced the evolution of the factorization models (e.g., PITF [204], FPMC [198]). In detail, BPR-MF [197] deserves particular attention because it boosted the MF research, and it is still considered as a state-of-the-art model. For completeness, Rendle also proposed Factorization Machines [196] that generalize the factorization approach. The biggest criticism of MF approaches, however, lies in their linearity. To address this concern, a recently popularized trend in the community of recommender systems is using deep neural architectures with deep neural networks that can model the non-linearity in data through nonlinear activation functions. In this respect, Neural Collaborative Filtering [113] and Neural Factorization Machines [108] have been recently proposed to overcome the inability of MF to capture non-linearities. Furthermore, Attentional Factorization Machines [273] use an attention network to learn the importance of feature interactions. Factorization models have been specialized for a variety of tasks such as Active-Learning [297], Context-aware [131], Cross-domain [84], Knowledge-aware [24, 26], and even explainable [290] recommendation.

In particular, Neural Collaborative Filtering [113] is one of the most representative recommendation approaches, which aims to estimate unknown user-item preference scores by exploiting deep neural networks [288]. Since Artificial Neural Networks (NNs) can approximate any continuous function on a compact set as long as the ANN has enough hidden states [67], He et al. [113] propose to exploit NNs to learn the affinity between p and q. Let $\Phi(\cdot)$ be the transformation function of the deep neural network defined as $\Phi : \mathbb{R}^{\dim(p)+\dim(q)} \to \mathbb{R}^d$, He et al. propose to concatenate the two

embeddings and predict the score as follows:

$$\psi^{\text{MLP}}(\mathbf{p}, \mathbf{q}) := \Phi([\,\mathbf{p}, \mathbf{q}]). \tag{4.1}$$

Additionally, He et al. defines a *generalized* matrix factorization model, in which p and q are combined using element-wise multiplication ($\odot$):

$$\psi^{\text{GMF}}(\mathbf{p}, \mathbf{q}) := \mathbf{p} \odot \mathbf{q}, \tag{4.2}$$

Finally, He et al. propose a comprehensive model, named NeuMF, that combines the two previous approaches together:

$$\psi^{\text{NeuMF}}(\mathbf{p}, \mathbf{q}) := \psi^{\text{MLP}}(\mathbf{p}, \mathbf{q}) + \psi^{\text{GMF}}(\mathbf{p}', \mathbf{q}'), \tag{4.3}$$

where the prime symbol ($'$) suggests that those embeddings might have a different size and are, in fact, different from the former ones. Finally, a careful reader may have noticed that ψ has an output dimension of d. This is correct, since He et al. applies a final prediction layer on top of them:

$$\phi^{\text{NCF}} := \sigma(\mathbf{W} \cdot \psi(\mathbf{p}, \mathbf{q})) \tag{4.4}$$

where σ is an activation function, and W is an additional weight matrix that is learned along with the other model parameters.

More recently, Rendle et al. [200] define the embeddings as model parameters, and the affinity between p and q is modeled by means of a dot product:

$$\phi^{\text{dot}}(\mathbf{p}, \mathbf{q}) := b_g + b_p + b_q + \sum_{f=1}^{d} p_f q_f, \tag{4.5}$$

where g_b, b_p, and b_q denote the global, user, and item bias, respectively. In this way, [200] presents a direct comparison between NNs and MF by changing the underlying operation between the embeddings, while keeping everything else comparable.

4.2.2 Settings

Although this study involves the replication of the results from three different studies, this paper mainly aims to replicate the results from Rendle et al. [200]. In Rendle

et al., the authors retrieve the already split datasets from the original NCF repository[1]. Specifically, He et al. [113] provide a split version of `MovieLens` and `Pinterest`. To split these well-known datasets, the authors adopt a temporal leave-one-out policy, moving the last user interaction into the test set. Furthermore, they binarize `MovieLens` to make the two datasets coherent with implicit feedback. Finally, they evaluate the methods on a shortlist of 101 candidate items for each user. This list comprises one relevant item (i.e., the transaction in the test set) and 100 negative items randomly sampled from not consumed items. In He et al. [113] and Rendle et al. [200], the authors evaluate the performance on top-10 recommendation lists computing Hit-Rate (HR) and Normalized Discounted Cumulative Gain (nDCG). The first estimates how many users have the withheld item in the top-10. The second measures the capability of the methods to rank the relevant item. In the following, the formulation of nDCG as presented in Krichene et al. [143] is adopted since it is the same one adopted in Rendle et al. [200]. Moreover, He et al. [113] and Rendle et al. [200] select the best models, for each recommendation system, according to HR@10. In this paper, the model selection follows the same strategy.

The present study involved the implementation of seven recommendation methods. MF implementation was designed accordingly to Rendle et al. [200] (also provided as a public repository[2]). Regarding NeuMF, the implementation refers to He et al. [113]. Finally, for the five remaining baselines, the implementation refers to Dacrema et al. [68] since it is the source for some of the results reported in Rendle et al. [200]. More specifically, the five implemented baselines are Slim [184], iALS [119], PureSVD [66], EASE^R [231], and RP$^3\beta$ [190]. According to the investigation provided by the authors [68], we replicate the baseline training exploiting the best hyperparameters found in the additional material[3].

4.2.3 Results

The first set of experiments aims to replicate Table 1 from Rendle et al. [200]. In that table, the authors compare Neural Matrix Factorization (NeuMF) and MF with a shortlist of baselines: Popularity, SLIM, and iALS. In detail, the authors report from Dacrema et al. [68] the results for Popularity, SLIM, NeuMF, and iALS. Instead, MF is trained using the publicly available implementation they provide. Overall, this

[1]https://github.com/hexiangnan/neural_collaborative_filtering

[2]https://github.com/google-research/google-research/tree/master/dot_vs_learned_similarity

[3]https://github.com/MaurizioFD/RecSys2019_DeepLearning_Evaluation/blob/master/DL_Evaluation_TOIS_Additional_material.pdf

Table 4.1 Comparison of NeuMF and MF with various baselines with cutoff@10. The table replicates (and compare with) the results from Dacrema et al. [68] and Rendle et al. [200]. The best results are highlighted in bold, the second best results is underlined. The columns with the Δ symbol indicate the absolute variation (for each metric) between the values of the experiments reproduced and those reported in the articles by Dacrema et al. [68] and Rendle et al. [200].

Algorithms	`MovieLens`		`MovieLens` [68, 200]		Δ `MovieLens`		`Pinterest`		`Pinterest` [68, 200]		Δ `Pinterest`	
	nDCG	HR	nDCG	HR	nDCG	HR	nDCG	HR	nDCG	HR	nDCG	HR
MostPop	0.2542	0.4535	0.2543	0.4535	$-1\cdot10^{-04}$	0	0.1410	0.2743	0.1409	0.2740	$1\cdot10^{-04}$	$3\cdot10^{-04}$
SLIM	0.4480	0.7164	0.4468	0.7162	$1.2\cdot10^{-03}$	$2\cdot10^{-04}$	0.5615	0.8696	0.5601	0.8679	$1.4\cdot10^{-03}$	$1.7\cdot10^{-03}$
iALS	0.4385	0.7123	0.4383	0.7111	$2\cdot10^{-04}$	$1.2\cdot10^{-03}$	0.5587	0.8766	0.5590	0.8762	$3\cdot10^{-04}$	$4\cdot10^{-04}$
NeuMF	0.4211	0.6952	0.4349	0.7093	$-1.38\cdot10^{-02}$	$-1.41\cdot10^{-02}$	0.5480	0.8704	0.5576	0.8777	$-9.6\cdot10^{-03}$	$-7.3\cdot10^{-03}$
MF	**0.4545**	**0.7310**	**0.4523**	**0.7294**	$2.2\cdot10^{-03}$	$1.6\cdot10^{-03}$	**0.5776**	**0.8898**	**0.5794**	**0.8895**	$1.8\cdot10^{-04}$	$3\cdot10^{-04}$
EASE^R	0.4494	0.7192	0.4494	0.7192	0	0	0.5605	0.8684	0.5604	0.8684	0	0
RP$^3\beta$	0.4011	0.6758	0.4011	0.6758	0	0	0.5685	0.8796	0.5685	0.8796	0	0
PureSVD	0.4299	0.6926	0.4303	0.6937	$-4\cdot10^{-04}$	$-1.1\cdot10^{-03}$	0.5233	0.8261	0.5241	0.8268	$-8\cdot10^{-04}$	$-7\cdot10^{-04}$

table questions the prominence of NeuMF and shows the high performance achieved by MF.

Hence, Table 4.1 replicates and extends the results provided in Table 1 from Rendle et al. [200]. In this study, all the recommendation algorithms have been retrained according to the best hyperparameters provided in Dacrema et al. [68]. Specifically, Table 4.1 reports HR and nDCG values for `MovieLens` and `Pinterest` datasets, respectively. The careful reader may have noticed that, for each dataset, both replicated and original results are reported. Original results columns are marked with references to the source papers. Dacrema et al. [68] also consider other recommendation algorithms. Interested in a more comprehensive comparison, we have selected EASE^R, RP$^3\beta$, and PureSVD for further replication. Finally, since Dacrema et al. [68] do not consider Matrix Factorization, no confusion arises regarding the origin of the results. Interestingly, Table 4.1 further confirms the findings of the original experiments showing that MF consistently overcomes the other baselines. It is worth mentioning how well the new experiments approximate the original ones.

Nonetheless, Rendle et al. [200] clearly state, in Table 1, that MF results are reported from (their) Figure 2. That figure compares MF, Learned Similarity (MLP), NeuMF, and pretrained NeuMF, considering different embedding sizes. However, the results reported from Rendle et al. (except for MF) are from He et al. [113]. Therefore, to conduct a thorough replication, we herein replicate some pivotal experiments reported in that figure. In detail, we have decided to replicate six MF experiments (three embedding sizes for each dataset) and eight NeuMF experiments (four embedding sizes for each dataset). For MF, we considered 32, 128, and 192 as embedding sizes. For

NeuMF, we considered 24, 48, 96, and 192 as embedding sizes (according to Appendix 3 from Rendle et al. [200]).

Thus, Figure 4.1 reports the original values from He et al. [113] regarding Learned Similarity (MLP) and pretrained NeuMF and reports our replicated experiments' results for MF and NeuMF. It is noteworthy mentioning that our MF experiments overlap with Rendle et al. showing that the MF curves dominate the others. However, the NeuMF curve shows different behavior from He et al. It is even more interesting to notice that the NeuMF experiment with 48 as the embedding size is also reported by Dacrema et al. [68], and the results are very close to ours.

Reviewing `Pinterest` results, MF confirms to be the best model in terms of HR and nDCG. Even in this context, NeuMF never overcomes the MF models. Actually, NeuMF reaches the best performance in terms of HR and nDCG for the NeuMF model with a number of factors equal to 16, according to the findings provided by Dacrema et al. [68]. Nonetheless, increasing that number of factors, we witness a performance decrease: both HR and nDCG decrease as the number of factors increases. Furthermore, Table 4.1 reports the overall results for the methods involved in the investigation. These outcomes confirm the evidence shown by Rendle et al. [200]: also other MF-based methods, like SLIM and iALS, outperform NeuMF. Beyond MF, also EASE^R provides a very notable performance. PureSVD behaves similarly to NeuMF. Finally, RP$^3\beta$ does not appear competitive as the other models in the investigation: its performance is consistently worse than the others. All these findings further confirm the results provided in Dacrema et al. [68]. Another finding (from Dacrema et al. [68]) the careful reader can rediscover in our experiments is the RP$^3\beta$ performance on the `Pinterest` dataset: although MF again demonstrates its higher accuracy, RP$^3\beta$ demonstrates competitive performance overcoming all the remaining baselines. Overall, the general take-home message of Dacrema et al. [68] experiments is confirmed: *NeuMF is often not better than relatively simple and well-known techniques.*

Finally, Table 4.2 compares, for the sake of completeness, our experiments on NeuMF without pretraining with the same configuration from He et al. [113]. The results in columns marked with the reference are from Table 2 in He et al. [113] and correspond to the results for the NeuMF model without pretraining. As shown before qualitatively, the replicated results obtained through the benchmark framework overlap the original ones. However, considering 64 factors on `Pinterest`, an appreciable difference can be observed that regards the nDCG value. This is probably due to the non-deterministic initialization of the model that leads to slightly different results. The effect seems to be more evident in the models with a greater embedding size, suggesting

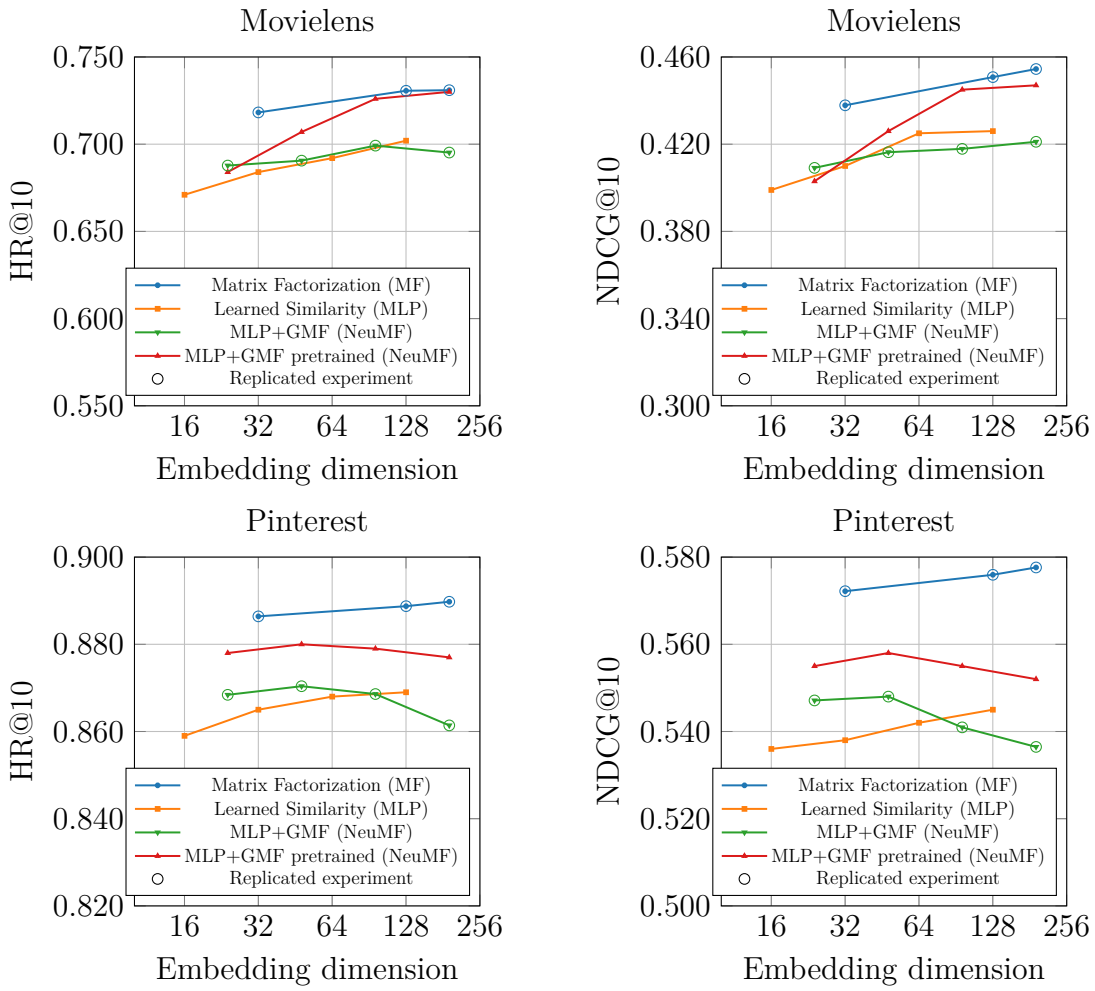


Fig. 4.1 Comparison of learned similarities (MLP, NeuMF) with a dot product: The results for MLP and pretrained NeuMF are from He et al. [113] and Rendle et al. [200]. MF substantially outperforms MF, NeuMF, and pretrained NeuMF. Nonetheless, on `MovieLens`, when considering large embeddings, pretrained NeuMF is competitive.

Table 4.2 Performance of NeuMF without pre-training. The table compares replicated experiments (on the left) with prior experiments He et al. [113]. Differently from He et al. [113], the results on `Pinterest` show a performance decrease with 64 factors. All metrics are with cutoff@10.

Factors	`MovieLens`		`MovieLens` [113]		`Pinterest`		`Pinterest` [113]	
	nDCG	**HR**	**nDCG**	**HR**	**nDCG**	**HR**	**nDCG**	**HR**
8 [113] - 24 [200]	0.409	0.688	0.410	0.688	0.547	0.868	0.546	0.869
16 [113] - 48 [200]	0.416	0.691	0.420	0.696	0.548	0.870	0.547	0.871
32 [113] - 96 [200]	0.418	0.699	0.425	0.701	0.541	0.869	0.549	0.870
64 [113] - 192 [200]	0.421	0.695	0.426	0.705	0.536	0.861	0.551	0.872

that the model accumulates the initial uncertainties. Remarkably, the deviation in the results exhibits a different trend (from the original model). Even though this could be a signal of lack of robustness of the model, further investigation is needed to shed light on this behavior.

An extended Accuracy evaluation

The existence of a high correlation between the accuracy metrics has been recently shown [249]. Nevertheless, an evaluation that examines only HR and nDCG could be quite limited. Therefore, we further extend the previous analysis considering other five metrics: F1-measure (F1) [98], Limited Area Under the Curve (LAUC) [223], Mean Average Precision (MAP) [98], Mean Average Recall (MAR) [98], and Mean Reciprocal Rank (MRR) [255].

Table 4.3 reports the results for the extended accuracy evaluation. Observing the big picture, MF is still one of the most competitive models, consistently being the best model regarding all the considered metrics on `Pinterest`. However, the situation is quite different on `MovieLens`, since EASE^R shows the best performance in terms of MAP, MAR, and MRR. Another interesting confirmation is the RP$^3\beta$ performance on `Pinterest`. For all the considered metrics, it shows to be the second-best model. However, if we observe the outcomes on `MovieLens`, the situation is much more confusing. Previous experiments showed that MF was the most accurate method, followed by EASE^R. Table 4.3 shows a quite different scenario, with EASE^R being the best model regarding MAP, MAR, and MRR, and the second best concerning the remaining metrics. Conversely, MF still shows competitive results, but regarding MAP and MRR, it is not in the first two places. Overall, MF, Slim, and iALS outperform NeuMF on these two datasets, hence confirming the most important finding of the previous experiment.

Table 4.3 Comparison of NeuMF and MF with various baselines on an extended set of accuracy metrics with cutoff@10. The best results are highlighted in bold, the second-best result is underlined.

Alghoritms	MovieLens					Pinterest				
	F1	**LAUC**	**MAP**	**MAR**	**MRR**	**F1**	**LAUC**	**MAP**	**MAR**	**MRR**
MostPop	0.0825	0.4531	0.0647	0.3072	0.1937	0.0499	0.2742	0.0341	0.1717	0.1009
SLIM	0.1303	0.7159	<u>0.1204</u>	0.5372	<u>0.3648</u>	0.1581	0.8694	0.1535	0.6757	0.4649
iALS	0.1295	0.7117	0.1172	0.5288	0.3537	0.1594	0.8764	0.1525	0.6786	0.4587
NeuMF	0.1264	0.6947	0.1120	0.5106	0.3363	0.1583	0.8702	0.1487	0.6653	0.4472
MF	**0.1316**	**0.7232**	0.1188	<u>0.5383</u>	0.3573	**0.1618**	**0.8896**	**0.1584**	**0.6958**	**0.4796**
$EASE^R$	<u>0.1308</u>	<u>0.7187</u>	**0.1210**	**0.54202**	**0.3655**	0.1579	0.8682	0.1532	0.6752	0.4639
$RP^3\beta$	0.1229	0.6753	0.1053	0.4853	0.3166	<u>0.1599</u>	<u>0.8794</u>	<u>0.1554</u>	<u>0.6836</u>	<u>0.4710</u>
PureSVD	0.1259	0.6921	0.1153	0.5178	0.3486	0.1502	0.8259	0.1422	0.6339	0.4286

Statistical hypothesis tests

To complete the study regarding the accuracy evaluation, we investigated whether the differences between the accuracy results of the various methods are statistically significant. Figure 4.2 shows eight heatmaps of statistical significance calculated with the Student's paired t-test. Statistically significant differences (with a p-value lower than 0.05) are drawn in green. In contrast, p-values greater than or equal to the 0.05 threshold value are colored by shades of red.

Figure 4.2 confirms that MF significantly overcomes the other methods regarding nDCG and HR on both `MovieLens` and `Pinterest`. Besides MF, and considering the same metrics, the differences between $EASE^R$, iALS, and Slim are not always statistically significant. Moreover, when analyzing MAP and MRR, it is noteworthy that the difference between $EASE^R$, Slim, and MF are not significant. For what concerns NeuMF, the situation is different. Indeed, the differences with $EASE^R$, PureSVD, and Slim are not always significant. Finally, $RP^3\beta$ deserves a concluding remark since all the differences with the other models are statistically significant, thus confirming its positive performance on `Pinterest` and the below-the-average one on `MovieLens`.

4.2.4 Novelty and Diversity

Once it is established how accurate the various methods are, our study expands beyond the accuracy evaluation. This section focuses on the ability of the recommendation algorithms to propose unknown items (Novelty), on overall item coverage, and on the ability to suggest highly diversified recommendation lists. For what concerns

Fig. 4.2 Statistical hypothesis tests using Student's paired t-test with a threshold value (light red) of p=0.05. Algorithm pairs which results are statistically significant are in green, the results that are not statistically significant are in red.

Novelty, we measure Expected Free Discovery (EFD) [251] and Expected Popularity Complement (EPC) [251], which measure the ability of a recommendation system to recommend items from the long tail. Concerning aggregate diversity metrics, we adopt Item Coverage [98] that measures the overall number of items the recommender suggests to the population. Finally, to measure how diversified the recommendation lists are, we exploit two popular distributional inequality metrics, the Gini Index (Gini) [98] and Shannon Entropy (SE) [98]. The Gini Index is defined as 1 - Gini Index from Gunawardana et al. [98], so that a higher value corresponds to a greater degree of diversification.

Figure 4.3 and figure 4.4 show twelve bar charts that compare MF and NeuMF with the other baselines regarding the six observed metrics on the two datasets. Let the analysis focus on Novelty. It is worth noticing that, even here, MF outperforms NeuMF and the other baselines since it generates recommendation lists with a larger number of items belonging to the long tail. Conversely, NeuMF shows poor performance, and only RP$^3\beta$ and Most Popular behave worse. In general, also other matrix factorization models such as iALS and SLIM are shown to be competitive against the other baselines under analysis. However, under the perspective of recommendation Diversity, the scenario dramatically changes. In fact, regarding Item Coverage, NeuMF is the best performing model on `MovieLens`, and a very competitive one on `Pinterest` (the best

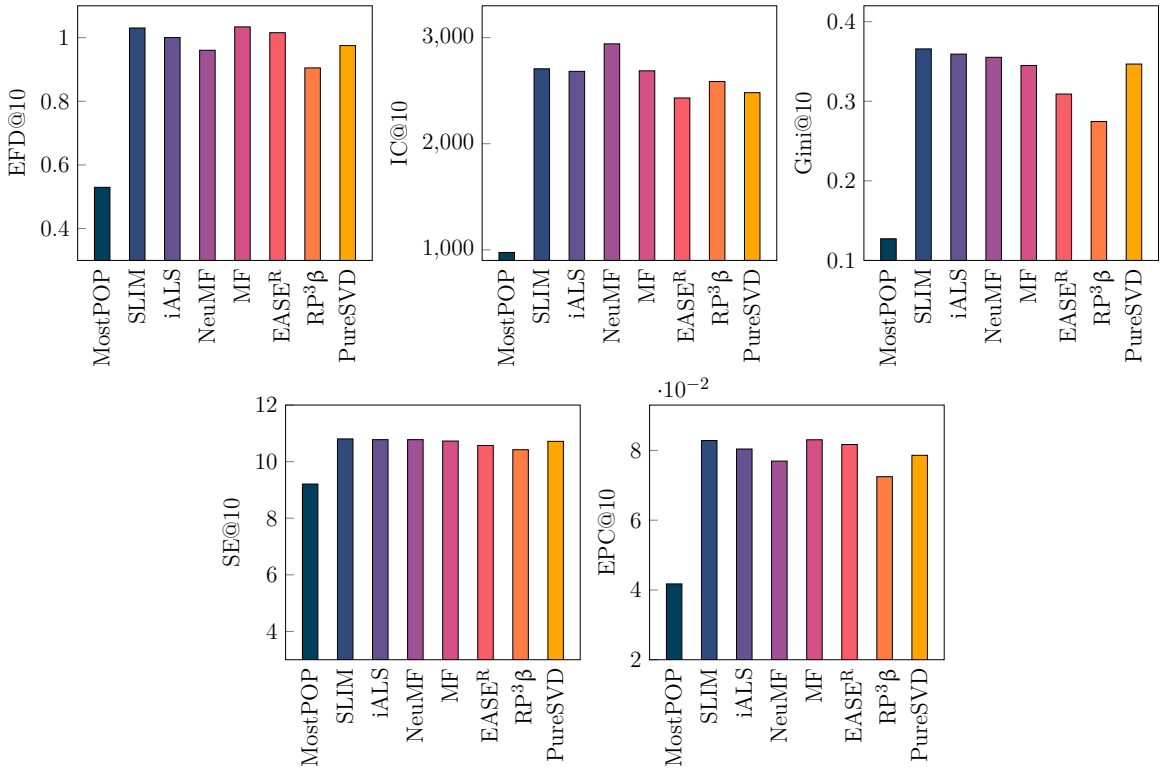


Fig. 4.3 Novelty and Diversity comparison of NeuMF and MF with various baselines (higher is better) for `MovieLens` dataset.

one is RP$^3\beta$), suggesting a higher overall number of items present in the catalog. Conversely, MF (and the other MF-based models) are not able to win the comparison. For what regards recommendation list diversification, the Gini bar chart reveals a more clear ranking of the methods. Again, MF fails to be effective in terms of diversity, and, on `MovieLens`, only EASE^R and RP$^3\beta$ show lower results. NeuMF shines neither on `MovieLens` dataset nor on `Pinterest` dataset. However, in both cases, it shows a greater propensity to generate personalized lists than MF. Interestingly, on both datasets, the iALS model is particularly competitive regarding the two distributional inequality metrics. Finally, even here, the reader may appreciate how different the RP$^3\beta$ performance is on the two datasets.

4.2.5 Analysis of Recommendation Biases

In the final part of the study, we focus on how the recommendation algorithms induce or amplify bias into the recommendation lists. Indeed, user-item interactions are often distributed unevenly over different groups of users and categories of items. This could be due to various reasons ranging from the naturally varying user preferences

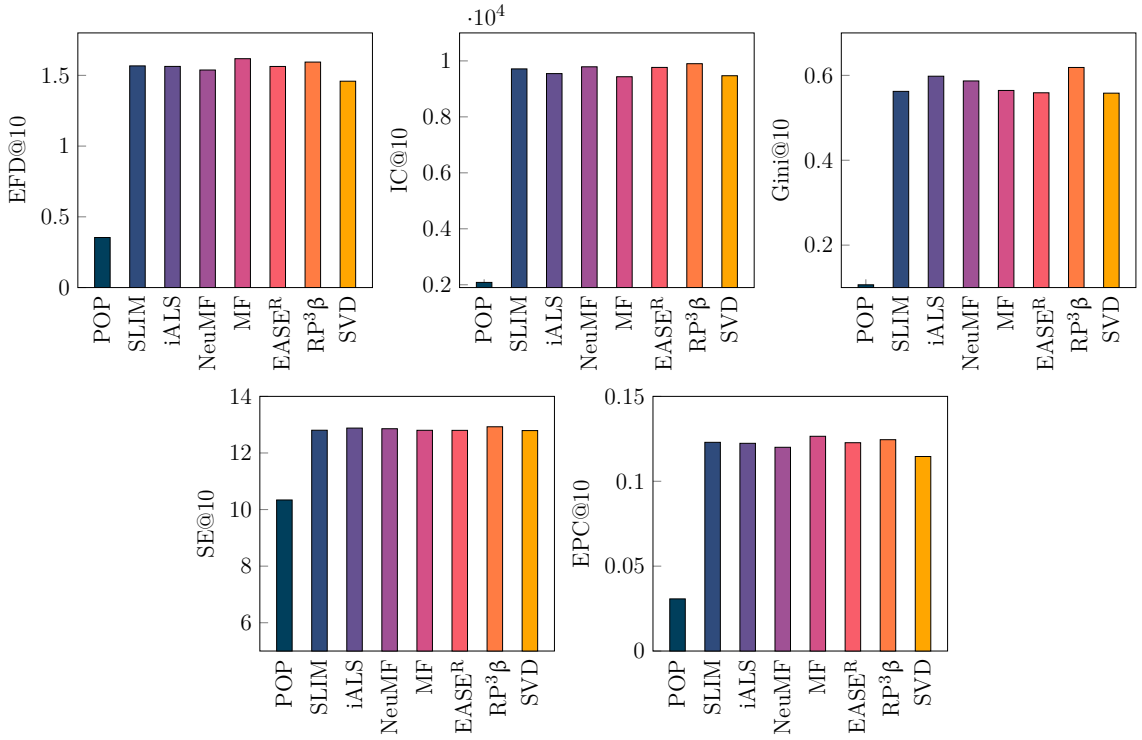


Fig. 4.4 Novelty and Diversity comparison of NeuMF and MF with various baselines (higher is better) for `Pinterest` dataset.

to the existence of a recommendation system in the preference collection system. Recommendation algorithms can inherit or even amplify this imbalanced distribution, leading to various kinds of bias. To examine the bias effect we consider five different metrics: Average Coverage of Long Tail items (ACLT) [6], Average Percentage of Long Tail Items (APLT) [5, 6], Average Recommendation Popularity (ARP) [6, 279], Ranking-based Statistical Parity (RSP) [300], and Ranking-based Equal-Opportunity (REO) [300].

Figure 4.5, and figure 4.6 show ten bar charts that compare MF and NeuMF against the other baselines regarding these five bias measures on the two datasets. The most straightforward metric to analyze is ARP. This metric measures the average popularity of the recommended items in each list. Interestingly, MF and NeuMF behave similarly on `MovieLens`, while EASE^R and RP$^3\beta$ are more prone to suggest popular items. In contrast, the other MF-based methods, iALS, Slim, and PureSVD, show the best performance. However, on `Pinterest`, the ranking is less clear since all the methods behave in a similar way. Let the analysis focus on ACLT and APLT. APLT measures the average percentage of long-tail items in the recommended lists, while ACLT measures how much exposure long-tail items get in the recommendations. These two metrics exhibit three interesting behaviors: (i) both iALS and NeuMF seem

to be less prone to these kinds of biases, (ii) MF, EASE^R, and PureSVD show to be heavily affected by them, (iii) the difference of RP$^3\beta$ performance on the datasets influences the bias of the generated recommendations.

Finally, we focus our investigation on RSP and REO. RSP measures whether items in different groups have the same probabilities of being recommended. Poor RSP means one or more groups have lower recommendation probabilities than others. REO measures the bias that items in one or more groups have lower recommendation probabilities given the items enjoyed by users. Differently from RSP, REO-based bias does not depend on sensitive attributes.

In this study, even though additional information could be retrieved to form item groups, the purpose is to conduct the investigation based on the same information available to the original authors. Therefore we formed two distinct groups of items based on the popularity signal. One group comprises the 20% most popular items, while the other includes the remaining items. For this reason, in the following, we refer to them as PopRSP and PopREO.

On `MovieLens` dataset, iALS and SLIM exhibit the best performance regarding both metrics. Even here, NeuMF demonstrates to be less prone than MF to this type of bias. MF does not show unsatisfactory results regarding both statistical parity and equal opportunity, but it never overcomes NeuMF. Finally, EASE^R, RP$^3\beta$, and PureSVD are affected by the bias and under-recommended items from minority groups, even though these items are present in the user history. In contrast, on `Pinterest`, RP$^3\beta$ shows leading performance, along with iALS and NeuMF. As detailed in Section 4.2.2, following He et al. [113] and Rendle et al. [200], all the recommenders were optimized for accuracy. It is left as future work an extended analysis where the effect of different optimization goals could have on the recommendation accuracy and beyond-accuracy dimensions, as in Kaminskas et al. [133].

4.2.6 Summary

Understanding how the different recommendation algorithms work under unique evaluation dimensions is critical to advance the field. In this work, we aimed to shed some light on this aspect, by contrasting recent models that are competitive against Neural Network approaches under complementary dimensions — not only accuracy, but novelty, diversity, coverage, and bias. In particular, we focus on the methods presented in He et al. [113] and Rendle et al. [200], and complemented our experimental exploration with the extensive analysis done in Dacrema et al. [68]. We have been able to replicate most of the results reported in those papers, where NeuMF is outperformed

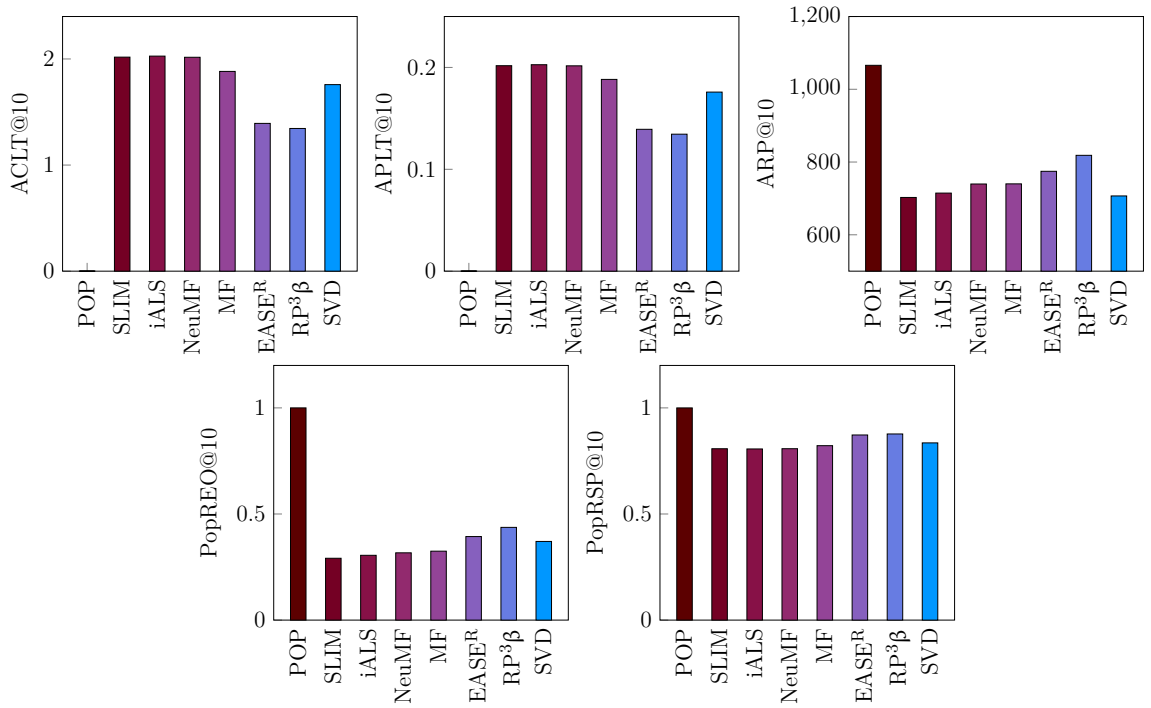


Fig. 4.5 Analysis of Bias for NeuMF, MF and various baselines on the `MovieLens` dataset. For ACLT, and APLT, higher is better; for ARP, PopREO, and PopRSP, smaller is better.

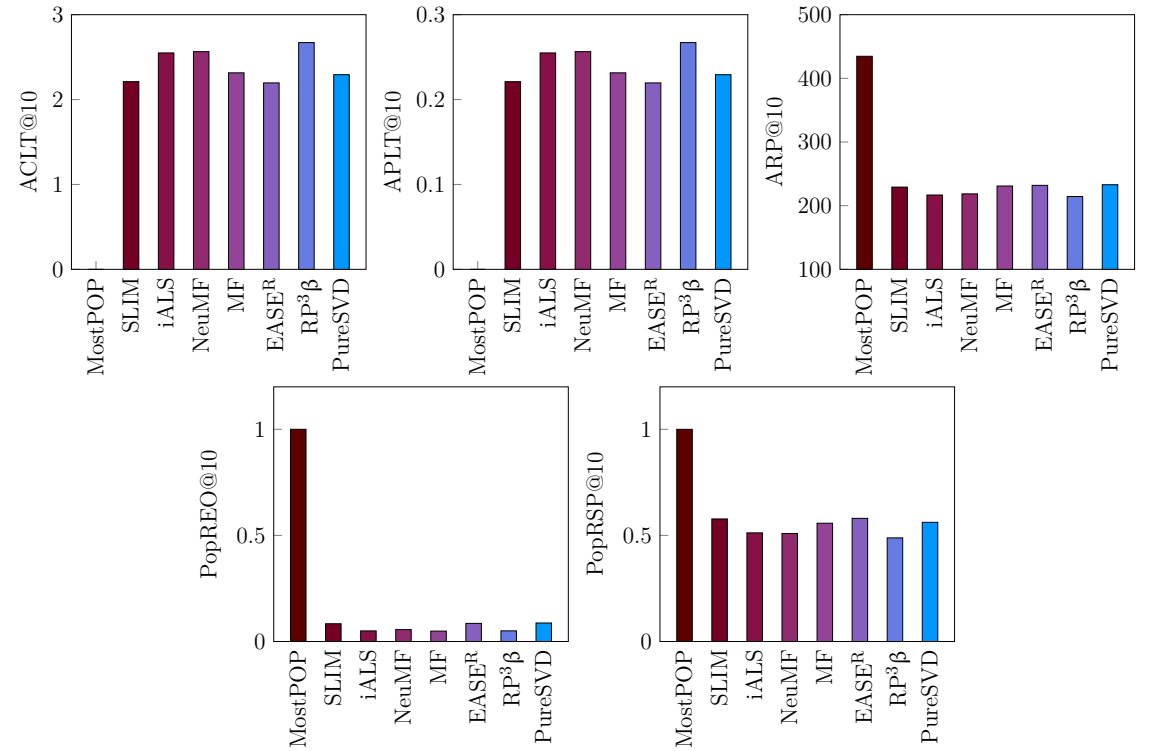


Fig. 4.6 Analysis of Bias for NeuMF, MF and various baselines on the `Pinterest` dataset. For ACLT, and APLT, higher is better; for ARP, PopREO, and PopRSP, smaller is better.

by the MF variation presented in Rendle et al. [200]. Moreover, when reproducing these approaches in new contexts, such as other evaluation dimensions or more accuracy metrics, baselines like EASE^R and RP$^3\beta$ are confirmed as solid candidates to be included in any comparison in the future, as their performance in terms of accuracy, diversity, and novelty is sometimes better than those of neural network approaches. However, it is important to highlight that the trend obtained for NeuMF is slightly different than from the original paper and, in particular, our extended analysis on accuracy evidences that the difference between this method and other baselines are not always statistically significant.

Our experiments have summarized and re-evaluated results from 3 recent papers, but they can be complemented in several ways. For example, one direction that has been unexplored so far is the effect that the splitting methodology or the item selection strategy could have in all these methods. Recent research has evidenced that how items are selected may affect the evaluation results [50]; however, because we wanted to replicate the exact conditions of these papers, we did not change these experimental settings. It will be interesting to analyze this aspect and how it (may) change the ranking of the methods. Another potential venue to improve this comparison is on the selection of datasets. Again, as we wanted to replicate the original papers, we were limited to use `MovieLens` and `Pinterest`, however, it is crucial to understand how these methods work in other domains and under a wide array of evaluation dimensions, such as those explored here.

4.3 Benchmarking Collaborative Filtering Recommendation models

Evaluation of the efficacy of a variety of various approaches is the primary focus of this contribution. In a style analogous to that which was carried out in Ferrari Dacrema et al. [85], we carefully consider a bunch of different collaborative filtering methods. These techniques include more classic approaches based on nearest-neighbors, different approaches to matrix factorization, linear models, as well as more recent methods based on deep learning.

4.3.1 Background

The purpose of this investigation is to analyze several algorithms using very *common* experimental conditions seen in the present body of research. This includes evaluating

them using very common datasets, evaluation criteria, and procedures. The following factors served as primary considerations in the process of selecting appropriate environments for the experiments described in this study. The work that Sun et al. [237] did, in which they methodically examined a variety of algorithms under a vast range of experimental setups, was the first source of motivation for our own research. Second, in order to pick certain experimental setups for the aim of our study, we looked through the most recent research to find examples of settings that are quite prevalent. Because of this, a number of more current models and simpler procedures that have been shown to be successful in recent research were included into Sun et al.'s study. Some of these approaches were not taken into account previously.

In a significant way, the overarching purpose of our current study diverges significantly from that of Sun et al. One of the primary goals of the research carried out by Sun et al. was to evaluate the degree to which certain features of the experimental approach, such as negative sampling, split-ratio, or dataset preprocessing, had an effect on the correctness of the results. On the other hand, the primary objective of our research is to provide a performance comparison of algorithms from several families for fairly typical experimental settings. Therefore, we have high hopes that our work will contribute to the establishment of a benchmark setting that is both universally accepted and continually updated. This setting will allow researchers to test their newly developed models against previously established models in a context that has been predetermined.

4.3.2 Settings

Datasets and Preprocessing

We discuss the findings that we acquired for three datasets that are often utilized in the most recent research: *MovieLens-1M*, *Amazon Digital Music*, and *Epinions*. All three of these datasets were collected from the internet:

- *MovieLens-1M* (`ML-1M`): The MovieLens datasets have been used extensively in the recommender systems literature for a significant amount of time [104], and several copies of the datasets can be found on the internet[4]. The `ML-1M` dataset was gathered on the MovieLens website between the years 2000 and 2003 and provides ratings for movies based on a scale of one to five. This dataset was used in our research. The dataset is very dense, and there are at least 20 ratings.

[4]https://grouplens.org/datasets/movielens/

- *Amazon Digital Music* (*AMZm*): This dataset was first developed in the context of image-based recommendation [175], and it is now included in a broader public collection of datasets[5] that was made available to the public. The data collection referred to as "Digital Music" includes reviews scraped from the Amazon website as well as ratings for each item on a scale from one to five.

- *Epinions*: The data for this dataset came from the now-defunct customer review website `epinions.com`[6], which was scraped in 2003. On the website Epinions, members were compensated according on how helpful their reviews were seen to be by other users. This was an unusual feature of the service. Because of this, Epinions has become more popular as a study tool for investigating confidence in recommender systems. The Epinions collection is comprised of two different datasets: the first one stores item ratings on a scale of one to five stars, while the second one compiles trust assertions made by users in a unary format. We would like to point out that in our research, rather than defining a specific (and in some ways arbitrary) threshold to binarize the rating dataset, we make use of the second dataset and consider the "trustable" people to be the target of the recommendation job.

It is important to emphasize that for the sake of our study, all three datasets are freely accessible to the public. Additionally, these datasets were chosen such that they represent a wide range of application areas associated with recommender systems. Other datasets, such as those from the Netflix Prize, were also quite popular for a period of time; but, in modern times, they are only utilized seldom, as shown by Liang et al. [156], and they are no longer publicly available to the public. In addition, in contrast to the competition for the Netflix Prize, *rating prediction* is no longer regarded as the most essential aspect of recommendation. Instead, the most prevalent objective these days is to determine the *ranks of the items*. As a further point of interest, the landscape is now being dominated by suggesting based on implicit feedback signals. This is due to the normal absence of explicit rating information present in a great number of apps. As a consequence of this, datasets that formerly included item ratings are often transformed into unary (like) signals. In our review, we also follow this method, and we transform the rating datasets from MovieLens and Amazon Digital Music to unary datasets by treating any rating over 3 as a positive signal[7]. In the case

[5]https://jmcauley.ucsd.edu/data/amazon/

[6]http://www.trustlet.org/epinions.html

[7]Alternative approaches exist in the literature for this conversion, e.g., considering every rating as positive in case it is higher than the user's average. Often, we also see that all ratings are converted

of the Epinions dataset, a conversion of this kind is not required since the data are already presented in the unary form.

In the real world, datasets are often relatively limited in scope. Therefore, another common pre-processing step in the literature is to create a more dense version of the datasets in order to ensure that there is a minimum number of interactions per user and item in the dataset, for example to allow for effective personalization. This can be done in order to ensure that there is a minimum number of interactions per user and item in the dataset. Because the datasets differed in so many ways, we came up with unique *p-cores* for each one. We verify that there are at least p interactions for each item and at least p interactions for each user in a p-core dataset. This ensures that the dataset is complete. In the context of our research, the generation of these p-core datasets was accomplished by an iterative process. During each iteration, the mentioned restrictions were applied until there were no longer any discernible changes in the dataset. For each of the datasets that were provided, a different value for p was determined by taking into account its size and density. While we utilized the most common value of p for Movielens 1M and Amazon Digital Music (p=10 for `ML-1M` and p=5 for *AMZm*, see Sun et al. [237]), we chose a p-core value for Epinions in order to get a similar density of the final matrix in comparison to the other two datasets. Due to the high sparsity of the dataset, in this particular instance, only a 2-core subset was used to calculate the results. Table 4.4 presents the results of the analysis in terms of the features of the dataset. It has come to our attention that doing away with unfavorable ratings and computing p-cores resulted in a sizeable decrease of the `ML-1M` dataset, and that doing the same thing for the *AMZm* dataset led to an even more significant reduction in the size of the dataset.

Table 4.4 Dataset characteristics before and after pre-processing

Dataset	**p-core**	**#interactions**	**#users**	**#items**	**#interactions**	**#users**	**#items**
		before pre-processing			*after pre-processing*		
Movielens 1M	10-core	1,000,209	6,040	3,706	571,531	5,949	2,810
Amazon Digital Music	5-core	1,584,082	840,372	456,992	145,523	14,354	10,027
Epinions	2-core	300,548	8,514	8,510†	300,475	8,485	8,463†

†: The Epinions dataset focuses on "trustable" user recommendation. Note that not all users are trustable candidates according to the historical transactions, which is why the number of users as recommendable items is lower than the number of users.

It is interesting to note that many of the datasets that are used now are not only over twenty years old but also rather tiny. For instance, the Netflix Prize dataset

to positive signals. This is however questionable as *(i)* a low rating, e.g., one star, is not a positive signal and *(ii)* it changes the problem into predicting who will rate what.

has 100 million ratings, but most of the datasets that are utilized today only have a few million ratings. Our working hypothesis is that the computational burden of several contemporary models discourages researchers from testing their hypotheses on more extensive data sets. The 20M version of the MovieLens datasets is one of the bigger *public* datasets, and it is occasionally utilized in the research that has been published [156]. However, the 1M version is used for assessments more commonly [237], which is the primary reason why we take into consideration in our research the 1M version. In addition, we found that systematically tweaking the hyperparameters for all of the datasets and models may be a computational challenge for certain models, even for the datasets of moderate size that are given in Table 4.4. This was something that we had seen.

Algorithms

Given the goals described above, we considered algorithms from different families in our analysis. All non-neural methods, except Bayesian Personalized Ranking (BPRMF) [197], were also considered as baselines in the recent analysis of recommendation algorithms presented in Ferrari Dacrema et al. [85]. Specifically, we considered the following techniques in our evaluation:

- Non-personalized baseline: Popularity-based recommendation (*MostPop*).
- Neighborhood-based and simple graph-based models: UserKNN [207], ItemKNN [220], RP$^3\beta$ [190].
- Linear models: SLIM [184], EASE^R [231].
- Matrix factorization models: BPRMF [197], MF2020 [199], iALS [119].
- Neural models: NeuMF [115], MultiVAE [156].

Table 4.5 provides more details for the compared algorithms and explains why we considered them for our study.

Evaluation Settings and Metrics

In this section, we provide details about the applied evaluation protocol, the evaluation metrics, and the hyperparameter tuning process.

[8]https://sifter.org/simon/journal/20061211.html

Table 4.5 Overview of compared algorithms

Family	**Algorithm**	**Description**
Non-personalized Baselines	MostPop	Recommends the most popular items to each user, where popularity is defined by the number of observed interactions in the training data.
	Random	Creates random recommendations for users. Mainly useful to provide a reference point for beyond-accuracy measures.
Neighbors and Graphs	UserKNN	A user-based nearest neighbor scheme proposed by Resnick et al. [207] in 1994 in an early paper on the GroupLens system. In general, we include early nearest-neighbor techniques here because *(i)* they let us gauge the progress on small datasets over time and *(ii)* they proved surprisingly effective in recent research [85].
	ItemKNN	Item-based nearest-neighbor algorithms were discussed in 2001 [220] and later successfully applied in industry around 2003 [159].
	$RP^3\beta$	This method ($RP^3\beta$) is a simple graph-based method [190] from 2017, which is conceptually similar to the ItemKNN method and can, despite its simplicity, lead to good performance [85].
Linear Models	SLIM	This regression-based method was proposed for top-n recommendation tasks in 2011 [184]. Like in a recent analysis [85], we use the *ElasticNet* version of the method [152], as it often leads to competitive results.
	$EASE^R$	Another linear model, proposed in 2019 [231], which works like a shallow autoencoder. We include this method because it often leads to good results despite its simplicity.
Matrix Factorization	MF2020	Matrix factorization methods were initially explored using Singular Value Decomposition in 1998 [42]. Later, in particular during and after the Netflix Prize, various machine learning approaches were proposed to learn latent factors[8]. A recent analysis shows that these methods from the late 2000s are still competitive. In our study, we use a very recent MF model from Rendle et al. [199] proposed in 2020, dubbed MF2020.
	iALS	This method from 2008 uses an Alternating Least Squares approach and is particularly designed to learn factor models for implicit feedback datasets [119]. The method is widely used as a non-neural baseline in the literature.
	BPRMF	This method from 2009 was also designed for implicit feedback and introduces a novel optimization criterion. We use the MF variant in our experiments, which is also frequently used as a non-neural baseline in the literature [197].
Neural Models	NeuMF	NeuMF was proposed in 2017 [115] and is an early and influential deep learning model used for recommendation. It generalizes matrix factorization and replaces the inner product with a neural architecture. The method is widely used as a neural baseline in the recent literature.
	MultiVAE	This model was designed for implicit feedback data, published in 2018, and is based on variational autoencoders [156]. According to the analysis in Ferrari Dacrema et al. [85], this method outperformed existing non-neural baselines in an independent evaluation.

Evaluation Protocol. We used a common *repeated* 80-20 hold-out splitting procedure in our experiments [199]. Correspondingly, each dataset is randomly split to sample chunks containing around 20% of the data. In each evaluation round, 20% of

the data are used for testing and the remaining 80% are for training. Each experiment is repeated five times. Later in Section 4.3.3, we report the mean of the observed values of the cross-validation runs.

We note that in the recent literature often only the results of one single training-test split are reported. While this data-splitting is typically done randomly in previous studies, we argue that cross-validation usually leads to more reliable results.

Metrics. We collect a rich variety of accuracy metrics as well as a number of "beyond-accuracy" measures that are commonly used in the literature to assess additional quality aspects of recommendation lists.

- In terms of *accuracy metrics*, we measure Normalized Discounted Cumulative Gain (nDCG), Mean Reciprocal Rank (MRR), Precision, Recall, Mean Average Precision (MAP), and F1 at common list lengths of 10, and 20. For F1, note that we compute it on a per-user basis and not simply as a harmonic mean of the averages of Precision and Recall across users. Thus, we have both metrics that take the position of the correct items into account and metrics that are agnostic of this aspect. Note here that we do not collect "sampled" metrics in our evaluation. In a *sampled metrics* approach, one test item is ranked within an often small list of "negative samples". Such a procedure, while widely used, was recently found to be unreliable [144]. Note that historically the majority of the literature considered error metrics (RMSE, MAE) for evaluation purposes. However, "*such classical error criteria do not really measure top-N performance*" [66]. Consequently, several ranking metrics have been proposed in the last two decades and were adopted to evaluate top-n recommendation tasks. The present work shows the evaluation results for the most commonly used ranking metrics.

- Considering *beyond-accuracy metrics*, we measured a broader range of metrics regarding popularity bias, novelty, fairness, and item coverage and concentration. The details of the considered metrics are provided in Table 4.6. We note that also the novelty and fairness metrics used here are based on popularity distributions of items. Specifically, for the PRSP and the PREO metrics, we consider the 20% most popular items as the "short head" and the rest as long-tail items.

- *Running times:* Modern machine learning models can be computationally expensive. Therefore, we measured the computation times required for each algorithm for training and testing.

Table 4.6 Overview of beyond-accuracy metrics

Aspect	Metric	Description
Coverage and Concentration	**IC**	Item Coverage (IC) measures how many items ever appear in the top-n recommendations of users.
	Gini	A measure of statistical dispersion, used to express the inequality of a distribution. A higher Gini index value ($\texttt{Gini} \in [0,\ldots,1]$) indicates a stronger concentration of the recommendations, e.g., on popular items [125]. To ease the interpretation of the results and associate higher values with better results in terms of non-concentrated recommendations, in Tables 4.12, 4.13, and 4.14 we report the value ($1-\texttt{Gini}$).
Novelty	**EFD**	Expected Free Discovery: A novelty measure proposed in [250] based on the *inverse collection frequency*. Like EPC, this measure expresses the ability of an algorithm to recommend relevant long-tail items.
	EPC	Expected Popularity Complement: This metric expresses the *expected "number of seen items not previously seen"* [250].
Fairness	**PREO**	The Popularity-based Ranking-based Equal Opportunity (REO) recommendation metric for assessing bias (fairness) was proposed in [300]. Lower values mean less biased recommendations.
	PRSP	Popularity-based Ranking-based Statistical Parity [300], to assess potential bias and thus fairness of the recommendations. Again, lower values mean less biased recommendations.
Popularity Bias	**APLT**	Average Popularity of Long-Tail Items: Measures the average popularity of long tail items in the top-n recommendations of users [6].
	ARP	Average Rating-based Popularity: This metric computes the popularity of the items in a recommendation list based on the number of interactions of each item in the training data [125].
	ACLT	Average Coverage of Long-Tail Items: Measures how many items from the long tail are covered in the top-n recommendations of users [6].

Hyperparameter tuning. We performed extensive hyperparameter tuning for all algorithms in our comparison, which is essential to understand what represents the *state-of-the-art*. Previous research [65] has identified that the lack of proper tuning of baseline algorithms may easily lead to a certain stagnation in the field, where new models are carefully tuned, whereas only limited effort sometimes goes into tuning existing baseline models.

For hyperparameter tuning, we relied on the HyperOpt library[9] and used Tree of Parzen Estimators (TPE) as an algorithm to find the best hyperparameters [40]. We determined suitable hyperparameter ranges for each algorithm from the literature, using, e.g., ranges that were earlier used in Ferrari Dacrema et al. [85] and other works. Depending on the number and ranges of the hyperparameters of each algorithms, we explored between 20 and 50 hyperparameter combinations for each model. Hyperparameter tuning was conducted on a validation set for each dataset, and nDCG@10 was used as an optimization target. As suggested by Anelli et al. [25], the nDCG metric

[9]http://hyperopt.github.io/hyperopt/

represents a reasonable choice for hyperparameter tuning. All hyperparameter ranges and the optimal values for each dataset and algorithm are reported in the provided online material for reproducibility.

4.3.3 Results

Accuracy Results

The results of the accuracy measurements for commonly used cutoffthresholds of 10 and 20 are shown in Table 4.7 (MovieLens-1M), Table 4.8 (Amazon Digital Music), and Table 4.9 (Epinions). The results for the cutoffthreshold of 50 are provided in the online material. We mark the best-performing method for each metric in bold font; the second-best result is underlined. The following main observations can be made.

- *Top-performing methods:* Considering nDCG as our main performance measure—most other metrics are correlated except for Recall in some situations—we find that the top three positions across all metrics and cutofflengths are taken by five algorithms: EASE^R, MF2020, SLIM, RP$^3\beta$, and, a bit surprisingly, UserKNN. Differences across the datasets exist, but the ranking at least at top places is quite consistent across the datasets. For `ML-1M`, EASE^R, MF2020, and SLIM are the best methods, whereas RP$^3\beta$, EASE^R, and SLIM are best for *AMZm*. These methods also work well in Epinions. For the Epinions dataset, however, UserKNN works even slightly better than EASE^R. Generally, the performance of the five top-performing methods is quite consistent, *with EASE^R always taking a top rank.* The MF2020 technique, in contrast, mainly seems to work particularly well for the dense ML1M dataset. We note here that UserKNN for the given datasets was always favorable over ItemKNN. It is noticeable that this evidence differs from some prior literature. In 2004 [74], it was suggested that item-based algorithms provide comparable or better quality recommendations than traditional user-neighborhood-based recommender systems. In 2011, researchers reported [184] that in their experiments item-based schemes outperform user-based ones. Similar observations were made in 2011 by Ekstrand et al. [82] for rating prediction tasks. In 2016, Christakopoulou et al. [61] generally assumed that the item-based methods had been shown to outperform the user-based schemes for the *top-n* recommendation task. In the analysis from 2021 [85], however, a general dominance of ItemKNN over UserKNN was not reported. There were cases where ItemKNN was better, but in the majority of the reported experiments UserKNN

was favorable, which suggests that the ranking of the methods may depend on dataset characteristics and specifics of the evaluation protocol.

- *Performance of neural methods:* The two neural methods considered here, NeuMF and MultiVAE, only led to medium performance on these datasets. While MultiVAE performed very well in an earlier comparison with traditional methods [85], we may assume that the modest size of the datasets might limit the power of this method in our experiment to a certain extent, see also the report on the use of deep learning methods at Netflix [232] or the discussions in Jannach et al. [126].

- *Fine-tuning opportunities:* The iALS and BPRMF methods often led to medium to modest performance in this comparison. Recent work indicates that further enhancing and fine-tuning methods like iALS for specific datasets may lead to additional performance improvements [203]. Note, however, that the goal of our work was to assess the performance of different algorithms under equal opportunities, i.e., by using a systematic but generic hyperparameter optimization procedure. Fine-tuning individual algorithms, e.g., by exploring rather uncommon ranges for the size of the latent factors, is of course possible, but not the focus of our work, which is about establishing a set of baselines (state-of-the-art) to consider in future works. Similar considerations apply for the neural methods, which may also be further tuned for individual datasets.

We note that the differences between the top-performing methods are sometimes small, often between one and a few percent. In papers that propose new models, we would therefore commonly expect statistical significance tests. For the evaluations reported in our study, we omit such tests as we have no prior hypotheses regarding which model would "win". Instead, the goal of our work is to provide guidance for researchers about which methods they might want to consider as baselines for comparison. We note that in many published papers no exact details are provided about how the significance tests are applied and prerequisites were validated. Also, in case of per-user comparisons of means, significance at common α-levels may be easy to achieve due to the large sample sizes [158].

Comparing our algorithm ranking with earlier works [85, 237], we find both commonalities and differences. A general commonality of these studies is that more traditional methods, including linear models, matrix factorization, or nearest neighbors frequently take the top positions of the rankings. For example, the innovative combination of Factorization Machines with BPR loss worked particularly well in [237]. Also SLIM and MF were in top positions for some datasets. Differently from our findings, NeuMF

Table 4.7 Accuracy Results for MovieLens-1M. The tables are sorted by nDCG in descending order. The notation *@N* indicates that the metrics are computed considering recommendation lists of N elements.

Algorithm	Top@10					
	nDCG	**MAP**	**MRR**	**Pre**	**Rec**	**F1**
EASE^R	**0.336**	0.335	**0.583**	0.274	**0.194**	0.190
SLIM	0.335	**0.337**	0.580	**0.275**	0.189	0.188
MF2020	0.329	0.327	0.563	0.272	0.190	**0.192**
UserKNN	0.315	0.314	0.554	0.256	0.183	0.179
RP$^3\beta$	0.315	0.313	0.556	0.256	0.184	0.179
iALS	0.306	0.304	0.542	0.252	0.179	0.176
MultiVAE	0.294	0.284	0.514	0.243	0.183	0.175
ItemKNN	0.292	0.293	0.518	0.242	0.163	0.163
NeuMF	0.277	0.275	0.494	0.232	0.157	0.158
BPRMF	0.275	0.271	0.502	0.226	0.166	0.161
MostPop	0.159	0.159	0.317	0.137	0.084	0.086
Random	0.008	0.007	0.020	0.007	0.004	0.004

Algorithm	Top@20					
	nDCG	**MAP**	**MRR**	**Pre**	**Rec**	**F1**
EASE^R	**0.335**	0.287	**0.587**	**0.216**	**0.289**	0.206
SLIM	0.332	**0.288**	0.584	**0.216**	0.283	0.204
MF2020	0.329	0.283	0.568	**0.216**	0.286	**0.207**
RP$^3\beta$	0.315	0.269	0.561	0.203	0.277	0.195
UserKNN	0.314	0.268	0.559	0.201	0.273	0.192
iALS	0.309	0.263	0.547	0.202	0.272	0.194
MultiVAE	0.304	0.250	0.519	0.199	0.281	0.195
ItemKNN	0.289	0.252	0.523	0.192	0.247	0.180
BPRMF	0.280	0.235	0.508	0.181	0.253	0.176
NeuMF	0.280	0.240	0.500	0.188	0.245	0.195
MostPop	0.161	0.141	0.326	0.114	0.137	0.103
Random	0.009	0.007	0.024	0.007	0.007	0.006

Table 4.8 Accuracy Results for Amazon Digital Music. The tables are sorted by nDCG in descending order.

Algorithm	Top@10					
	nDCG	**MAP**	**MRR**	**Pre**	**Rec**	**F1**
RP$^3\beta$	**0.085**	**0.040**	**0.115**	**0.023**	0.104	**0.036**
EASE^R	0.083	0.038	0.108	**0.023**	**0.106**	0.035
SLIM	0.081	0.037	0.106	0.022	0.104	0.035
UserKNN	0.081	0.037	0.105	0.022	0.104	0.035
iALS	0.073	0.032	0.093	0.021	0.099	0.033
ItemKNN	0.071	0.033	0.095	0.018	0.085	0.029
MF2020	0.057	0.024	0.067	0.017	0.083	0.028
NeuMF	0.056	0.024	0.068	0.015	0.074	0.025
MultiVAE	0.054	0.023	0.062	0.016	0.077	0.025
BPRMF	0.020	0.008	0.023	0.007	0.031	0.010
MostPop	0.012	0.005	0.016	0.004	0.016	0.006
Random	0.000	0.000	0.000	0.000	0.001	0.000

Algorithm	Top@20					
	nDCG	**MAP**	**MRR**	**Pre**	**Rec**	**F1**
RP$^3\beta$	**0.094**	**0.029**	**0.118**	**0.015**	0.132	**0.026**
EASE^R	0.092	0.028	0.111	**0.015**	**0.136**	**0.026**
SLIM	0.090	0.027	0.109	0.014	0.134	0.025
UserKNN	0.090	0.027	0.108	0.014	0.133	0.025
iALS	0.084	0.025	0.096	0.014	0.132	0.025
ItemKNN	0.078	0.023	0.098	0.012	0.108	0.021
MF2020	0.067	0.019	0.071	0.012	0.116	0.022
NeuMF	0.063	0.018	0.071	0.010	0.096	0.019
MultiVAE	0.062	0.017	0.066	0.011	0.105	0.019
BPRMF	0.025	0.007	0.025	0.005	0.047	0.009
MostPop	0.014	0.004	0.017	0.003	0.025	0.005
Random	0.001	0.000	0.001	0.000	0.001	0.000

more often worked very well for some of the datasets examined in Sun et al. [237]. A competitive performance of NeuMF was also observed in Ferrari Dacrema et al. [85], where it was, however, usually slightly outperformed by various non-neural methods. These differences may be attributed to different causes, including specifics of data-

Table 4.9 Accuracy Results for Epinions. The tables are sorted by nDCG in descending order.

Algorithm	Top@10					
	nDCG	**MAP**	**MRR**	**Pre**	**Rec**	**F1**
UserKNN	**0.164**	0.131	0.266	0.157	0.102	**0.100**
EASE^R	**0.164**	**0.132**	**0.268**	0.154	**0.104**	**0.100**
RP$^3\beta$	0.163	0.129	0.260	**0.159**	0.102	**0.100**
SLIM	0.156	0.126	0.254	0.148	0.101	0.096
MultiVAE	0.149	0.116	0.240	0.148	0.094	0.094
ItemKNN	0.138	0.111	0.224	0.133	0.090	0.087
MF2020	0.125	0.104	0.219	0.114	0.084	0.079
NeuMF	0.118	0.098	0.206	0.108	0.080	0.075
BPRMF	0.113	0.093	0.200	0.106	0.076	0.071
iALS	0.110	0.091	0.192	0.101	0.074	0.084
MostPop	0.045	0.037	0.093	0.036	0.029	0.025
Random	0.001	0.001	0.003	0.001	0.001	0.001

Algorithm	Top@20					
	nDCG	**MAP**	**MRR**	**Pre**	**Rec**	**F1**
UserKNN	**0.178**	0.108	0.272	0.076	**0.219**	0.093
EASE^R	0.177	**0.110**	**0.274**	**0.078**	0.216	**0.094**
RP$^3\beta$	0.176	0.107	0.266	0.075	0.218	0.092
Slim	0.170	0.106	0.260	0.076	0.210	0.091
MultiVAE	0.165	0.099	0.247	0.072	0.212	0.089
ItemKNN	0.151	0.094	0.230	0.068	0.190	0.084
MF2020	0.138	0.088	0.225	0.065	0.170	0.079
NeuMF	0.131	0.084	0.212	0.063	0.162	0.075
BPRMF	0.126	0.079	0.207	0.060	0.160	0.072
iALS	0.121	0.077	0.198	0.057	0.149	0.079
MostPop	0.052	0.032	0.100	0.025	0.061	0.029
Random	0.002	0.001	0.003	0.001	0.002	0.001

preprocessing and the evaluation procedures[10]. Differently from many earlier works, we apply cross-validation and compute *p-cores* iteratively instead of only filtering "cold" users and items once. Moreover, for some algorithms we explore a larger number of hyperparameter optimization trials than was done in some earlier works.

Finally, to obtain an overall picture of our accuracy results, we applied a Borda count *ranked voting* scheme to aggregate the outcomes of our experiments. To that purpose, we consider each observed ranking for each dataset and metric as a vote. When applying the original Borda count scheme, each candidate (i.e., algorithm) receives more points when it is placed higher in the ranking. In our lists of 12 candidates, the first candidate receives 11 points and the last-ranked candidate 0 points. Applying this scheme across all accuracy measures at list length 10 leads to the ranking shown in Table 4.10a.[11]

We emphasize that such a rank-based aggregation should be interpreted with great care as it might, for example, favor methods that work particularly well on a set of correlated metrics.

In agreement with the analysis presented by Valcarce et al. [248], we observed high correlation between ranking metrics and for the same metric using different cutoffs. For

[10]In the original paper proposing NeuMF, the authors for example used a *leave-one-out* procedure where only the last item of each user was retained in the test set [115].

[11]The maximum possible value for a method in Table 4.10a is 198, as we rank 12 algorithms according to 6 metrics for 3 datasets; 198=(12-1) × 1 × 3. For Table 4.10b and Table 4.10c, the maximum is correspondingly 33. Although Tables 4.7 to 4.9 report rounded values for the sake of clarity, rankings are assessed considering exact metric values.

Table 4.10 Algorithm ranking based on Borda count at cutofflength 10.

Rank	Algorithm	Count
1	$EASE^R$	185
2	$RP^3\beta$	169
3	SLIM	160
4	UserKNN	154
5	MF2020	115
6	ItemKNN	99
7	MultiVAE	92
8	iALS	90
9	NeuMF	61
10	BPRMF	45
11	MostPop	18
12	Random	0

(a) Overall

Rank	Algorithm	Count
1	$EASE^R$	31
2	UserKNN	27
3	RP3beta	27
4	SLIM	27
5	MF2020	19
6	ItemKNN	16
7	MultiVAE	15
8	iALS	13
9	NeuMF	12
10	BPRMF	7
11	MostPop	3
12	Random	0

(b) nDCG

Rank	Algorithm	Count
1	$EASE^R$	31
2	$RP^3\beta$	29
3	SLIM	26
4	UserKNN	25
5	MF2020	20
6	MultiVAE	17
7	ItemKNN	15
8	iALS	14
9	NeuMF	9
10	BPRMF	9
11	MostPop	3
12	Random	0

(c) Recall

example, in that work, when computing the correlation between cutoffs ranging from 5 to 100, the lowest one was 0.9, which still represents a very strong correlation. Because of this, we only considered one threshold for the measurement shown in Table 4.10a. Another known limitation of the Borda count scheme is that the ranking might change if a candidate is removed from the lists. Despite these limitations, we believe that the Borda count may represent a helpful summarization approach for the experiments in this paper. More fine-grained applications of the Borda count are possible as well to account for such correlations. In Table 4.10b and Table 4.10c, we report the Borda count rankings when considering only one specific measure, nDCG@10 and Recall@10, respectively. We select Recall as an example here, because all other metrics are usually more correlated with nDCG than Recall. The analysis in Table 4.10c actually shows that $RP^3\beta$ and SLIM work particularly well for Recall and are ranked higher than UserKNN for this metric.

Beyond-Accuracy Results

Table 4.12 shows the beyond-accuracy metrics results for the MovieLens dataset for the top-10 and top-20 recommendations[12]. The rows in the table are again sorted by accuracy (nDCG). We highlight the best values for each metric, not considering the Random and MostPop baselines, which only serve as reference points. Recommending random items will, for example, lead to high item coverage, but not to many relevant item suggestions.

[12]Detailed results for other datasets and cutoffthresholds can be found in the online material.

Table 4.11 Summary of Metric Correlations. A ✓ in a cell indicates a correlation of more than 0.9 (or beyond -0.9 vice versa) for one of the datasets. Two or three ✓ symbols mean that such a high correlation was also found for the second or the third dataset.

PPMCC	Gini	EFD	EPC	PREO	PRSP	ACLT	APLT	ARP
IC	✓	–	–	✓	–	–	–	✓
Gini		–	–	✓	✓✓	✓	✓	–
EFD			✓	–	✓	✓	✓	–
EPC				✓	–	✓	✓	–
PREO					✓	✓	✓	–
PRSP						✓✓✓	✓✓✓	–
ACLT							✓✓✓	–
APLT								–

In our analysis we found that some of our beyond-accuracy can be highly correlated, which is to some extent expected as many of them are based on item popularity characteristics, as discussed above. Table 4.11 shows the outcomes of an analysis of metric correlations. In this table, we report how many cases (datasets) a metric is correlated with another one with a Pearson product-moment correlation coefficient (PPMCC) higher than 0.9 or lower than -0.9. We can observe that both the ACLT and the PRSP metrics correlate consistently with the APLT metric. For the sake of conciseness, we therefore only report the APLT metric here and omit ACLT and PRSP from the tables.

Generally, we observe that the ranking of the algorithms is not entirely consistent across the datasets. Here, we summarize a number of patterns that we observed, having in mind that beyond-accuracy measures are only of secondary interest in this study.

- For **ARP**, which reports the average item popularity in the top-n lists, we find that BPRMF often has the strongest tendency to recommend popular items on all datasets. MF2020 and EASE^R are also often at the higher end regarding the popularity bias. The ranking of the algorithms, however varies across datasets. The differences between algorithms on the `ML-1M` dataset are also generally smaller than for other datasets. On the other end of the spectrum, we observe that the neural methods NeuMF and MultiVAE sometimes succeed in including less popular items in the recommendation lists. RP$^3\beta$ and ItemKNN are similarly successful on the Epinions and *AMZm* in this respect. The **APLT** metric, which considers the popularity and coverage of long-tail items are negatively correlated with the **ARP** metric, i.e., the more popular items are recommended, the fewer from the long tail.

Table 4.12 Beyond Accuracy Results for MovieLens-1M. The tables are sorted by nDCG in descending order. The notation *@N* indicates that the metrics are computed considering recommendation lists of N elements. To ease the interpretation of the results and to associate higher values with more diversified recommendation lists, we report the value of $1 - Gini\dagger$.

Algorithm	**Top@10**						
	IC	**Gini**	**EFD**	**EPC**	**PREO**	**APLT**	**ARP**
EASE^R	838.0	0.068	**2.690**	**0.583**	0.978	0.003	1,062.73
SLIM	654.2	0.052	2.672	0.244	0.995	0.001	1,121.38
MF2020	920.2	0.077	2.672	0.244	0.968	0.005	1,042.37
UserKNN	1075.2	0.067	2.489	0.227	0.971	0.010	1,085.55
RP$^3\beta$	854.4	0.048	2.461	0.223	0.959	0.011	1,181.64
iALS	712.2	0.080	2.516	0.232	0.997	0.000	935.91
M-VAE	**1625.2**	**0.136**	2.422	0.221	**0.828**	**0.042**	**871.87**
ItemKNN	1054.8	0.066	2.346	0.214	0.952	0.011	1,090.93
NeuMF	1367.2	0.111	2.292	0.209	0.910	0.028	938.86
BPRMF	1137.8	0.091	2.226	0.203	0.928	0.010	1,047.23
MostPop	56.2	0.005	1.187	0.103	1.000	0.000	1,746.69
Random	2810.0	0.876	0.074	0.006	0.039	0.696	151.05

Algorithm	**Top@20**						
	IC	**Gini**	**EFD**	**EPC**	**PREO**	**APLT**	**ARP**
EASE^R	1093.0	0.091	**2.264**	**0.207**	0.963	0.006	949.86
SLIM	854.0	0.069	2.239	0.205	0.986	0.003	1,017.37
MF2020	1128.8	0.095	2.257	**0.207**	0.946	0.009	969.26
RP$^3\beta$	1207.2	0.073	2.085	0.190	0.937	0.018	1,024.41
UserKNN	1465.8	0.092	2.090	0.191	0.947	0.017	970.26
iALS	901.000	0.105	2.138	0.197	0.983	0.002	838.66
M-VAE	**1924.8**	**0.156**	2.082	0.190	**0.792**	**0.052**	**821.85**
ItemKNN	1346.0	0.082	1.977	0.181	0.939	0.015	1,006.98
BPRMF	1386.2	0.111	1.893	0.173	0.890	0.017	968.78
NeuMF	1679.4	0.135	1.965	0.179	0.867	0.038	868.45
MostPop	92.0	0.010	1.039	0.092	1.000	0.000	1,570.67
Random	2810.0	0.911	0.075	0.007	0.037	0.693	151.35

- The novelty metrics **EPC** and **EFD**, like all remaining beyond-accuracy metrics considered here, are generally negatively correlated with the **ARP** metric as well. An interesting pattern here is that models that perform well on the nDCG are also mostly highly ranked in terms of the novelty metrics.

- Looking at the fairness metric **PREO**, which is also based on item popularity and where lower values are better, the picture is not so clear. The neural MultiVAE method, for example, seems to rather consistently produce relatively fair recommendations according to this metric. ItemKNN leads to very good

Table 4.13 Beyond Accuracy Results for Amazon Digital Music. The tables are sorted by nDCG in descending order.

Algorithm	Top@10						
	IC	**Gini**	**EFD**	**EPC**	**PREO**	**APLT**	**ARP**
RP$^3\beta$	**10016.0**	**0.609**	**0.293**	**0.024**	0.318	0.299	22.35
EASE^R	9441.0	0.233	0.267	0.022	0.542	0.071	47.99
SLIM	9659.4	0.253	0.263	0.022	0.548	0.086	43.34
UserKNN	9294.2	0.237	0.264	0.022	0.543	0.073	45.83
iALS	5941.6	0.177	0.243	0.020	0.700	0.015	37.52
ItemKNN	9976.2	0.528	0.247	0.019	**0.121**	**0.546**	**9.87**
MF2020	6389.4	0.135	0.187	0.016	0.661	0.014	51.48
NeuMF	8533.2	0.266	0.180	0.015	0.468	0.075	27.42
MultiVAE	9161.4	0.329	0.178	0.015	0.519	0.094	33.59
BPRMF	4283.0	0.034	0.064	0.006	0.787	0.002	108.30
POP	29.2	0.002	0.033	0.004	1.000	0.000	148.84
Random	10025.8	0.895	0.002	0.000	0.158	0.460	9.84

Algorithm	Top@20						
	IC	**Gini**	**EFD**	**EPC**	**PREO**	**APLT**	**ARP**
RP$^3\beta$	**9959.0**	**0.542**	**0.409**	**0.033**	0.308	0.299	23.76
EASE^R	7789.0	0.178	0.368	0.031	0.537	0.054	56.16
SLIM	8215.4	0.197	0.361	0.030	0.552	0.067	49.29
UserKNN	7703.8	0.181	0.363	0.030	0.552	0.056	51.91
iALS	4516.2	0.136	0.325	0.027	0.766	0.009	41.94
ItemKNN	9686.2	0.478	0.345	0.027	**0.097**	**0.550**	**9.88**
MF2020	4722.8	0.099	0.242	0.021	0.687	0.009	59.95
NeuMF	7365.2	0.228	0.245	0.020	0.455	0.058	30.24
MultiVAE	6043.0	0.189	0.235	0.020	0.578	0.045	40.48
BPRMF	3050.0	0.024	0.078	0.007	0.784	0.001	130.81
POP	15.6	0.001	0.039	0.004	1.000	0.000	182.80
Random	10025.8	0.852	0.002	0.000	0.229	0.460	9.84

results on the Epinions and Amazon dataset, and to average performance on the `ML-1M` dataset. For this latter dataset, the spread of values is however not too high.

- Finally, considering the **Gini** index, MultiVAE generally leads to lower concentration levels on `ML-1M`, and ItemKNN and RP$^3\beta$ have lower concentration effects for the Epinions and *AMZm* datasets. Looking at **Item Coverage**, both nearest-neighbor methods and the neural approaches are typically better than the matrix factorization techniques iALS and BPRMF. The patterns are however

Table 4.14 Beyond Accuracy Results for Epinions. The tables are sorted by nDCG in descending order.

Algorithm	Top@10						
	IC	**Gini**	**EFD**	**EPC**	**PREO**	**APLT**	**ARP**
UserKNN	3402.2	0.073	**1.198**	0.112	0.398	0.080	315.724
EASER	2765.0	0.055	1.197	**0.114**	0.501	0.046	341.486
RP$^3\beta$	**6009.0**	**0.197**	1.237	0.112	0.198	**0.275**	226.165
SLIM	3361.0	0.081	1.197	0.110	0.452	0.061	246.216
M-VAE	3386.8	0.089	1.105	0.102	0.364	0.105	293.641
ItemKNN	5832.8	0.160	1.084	0.097	**0.171**	0.267	**214.681**
MF2020	1235.2	0.028	0.921	0.090	0.786	0.008	368.003
NeuMF	3595.0	0.084	0.905	0.085	0.495	0.096	322.865
BPRMF	1322.8	0.018	0.824	0.081	0.651	0.012	475.629
iALS	1613.4	0.064	0.891	0.081	0.707	0.010	214.989
MostPop	41.6	0.001	0.273	0.030	1.000	0.000	719.301
Random	8443.0	0.823	0.011	0.001	0.142	0.738	27.565

Algorithm	Top@20						
	IC	**Gini**	**EFD**	**EPC**	**PREO**	**APLT**	**ARP**
UserKNN	4594.6	0.095	0.962	0.090	0.442	0.083	275.265
EASER	3705.0	0.071	**0.965**	**0.091**	0.522	0.047	297.062
RP$^3\beta$	7010.8	**0.232**	0.983	0.089	0.262	**0.268**	**198.240**
SLIM	4401.2	0.098	**0.965**	0.089	0.493	0.063	228.155
M-VAE	4249.6	0.112	0.900	0.083	0.426	0.112	255.457
ItemKNN	**7100.2**	0.188	0.875	0.079	**0.272**	0.262	199.488
MF2020	1631.2	0.040	0.766	0.074	0.776	0.009	320.867
NeuMF	4647.0	0.108	0.756	0.071	0.542	0.105	276.817
BPRMF	1793.6	0.026	0.693	0.067	0.640	0.012	409.832
iALS	2052.0	0.078	0.730	0.066	0.654	0.016	202.718
MostPop	72.0	0.002	0.244	0.027	1.000	0.000	629.745
Random	8443.6	0.875	0.011	0.001	0.079	0.738	27.654

not consistent across datasets. EASE^R, for example, leads to relatively high item coverage on *AMZm*, but not on the other datasets.

Overall, not many consistent patterns regarding beyond-accuracy measures across all three datasets can be observed. One example of such a pattern is a certain popularity bias of the BPRMF method, which was previously observed [125]. Some patterns, like good item coverage for ItemKNN, are only found for the *AMZm* and Epinions datasets, which suggests that the widely used `ML-1M` dataset may be to some extent unique and it stands to question how representative this dense dataset is for other typical application scenarios, e.g., for e-commerce settings.

Time Measurements

We carried out all experiments on a computing cluster of our organization. The used cluster is based on IBM Power9 processors and has 980 nodes. Each node is equipped with 32 cores and 4 NVIDIA Volta GPUs. One cluster node with 200GB of RAM with 4 logical CPUs was reserved for each experiment. In addition, one NVIDIA Volta GPU with 16GB of RAM has been allocated for the experiments with the neural models NeuMF and MultiVAE. Table 4.15 shows the time measurements obtained for the three datasets, using the optimal parameters (e.g., number of latent factors) that were determined through hyperparameter tuning. The numbers reported in the table refer to the time needed (in seconds) to train the model once, and to create and evaluate the recommendation lists for all users in the test set.

Table 4.15 Training and evaluation time

Algorithm	**Time (s)**
MF2020	1.53×10^4
NeuMF	7.97×10^3
BPRMF	3.97×10^3
iALS	331.93
UserKNN	87.29
EASE^R	85.93
SLIM	73.19
MultiVAE	67.03
RP$^3\beta$	47.06
ItemKNN	42.74
Random	27.49
MostPop	24.63

(a) MovieLens-1M

Algorithm	**Time (s)**
NeuMF	3.57×10^4
iALS	2.90×10^4
MF2020	2.65×10^4
MultiVAE	1.37×10^4
EASE^R	1.85×10^3
BPRMF	1.51×10^3
SLIM	403.29
RP$^3\beta$	270.78
ItemKNN	257.96
UserKNN	247.68
Random	50.86
MostPop	45.58

(b) Amazon Digital Music

Algorithm	**Time (s)**
iALS	3.27×10^4
MF2020	1.97×10^4
NeuMF	3.46×10^3
BPRMF	2.26×10^3
EASE^R	1.12×10^3
SLIM	344.69
MultiVAE	215.43
RP$^3\beta$	148.24
UserKNN	144.05
ItemKNN	139.42
Random	47.99
MostPop	44.24

(c) Epinions

The results show that there is a substantial spread between the algorithms. While there are some models that complete training and testing within one minute, training the MF2020 method on the ML1M dataset, where it performed well, took several days. We note here that more efficient implementations of matrix factorization techniques have been proposed [202]. Also the NeuMF model needed substantial time to complete the computations. In contrast, the MultiVAE model, which was also originally evaluated on larger datasets in Liang et al. [156] was among the fastest models. The neighborhood-based models and RP$^3\beta$ were also implemented for high efficiency. For the other datasets, Epinions and Amazon, the results are similar with NeuMF and the matrix factorization models often taking substantial computation time. For this latter class of models, the efficiency also largely depends on the optimal number of latent factors.

Generally, combining the timing results with accuracy results from above, we see no clear indication for the given datasets that computationally more complex models are favorable in terms of prediction accuracy.

4.3.4 Summary

In recent years, several researchers have identified major challenges with respect to reproducibility and progress in recommender systems research. Various factors contribute to these phenomena, in particular *(a)* that a larger fraction of published research is not reproducible because authors do not share the required artifacts and *(b)* that the experiments in published research mainly aim to highlight the superiority of a new model. In the context of this latter aspect, this practically often means that only the new method is carefully fine-tuned but not the compared baseline methods. Furthermore, the choice of the baselines is sometimes limited to very recent models, thus probably missing strong baselines that were published earlier.

With this work, our goal is to address these open issues in different ways. First, we conducted a large number of reproducible experiments on different datasets and involving a variety of algorithms from different families in order to provide an independent evaluation of existing techniques along different quality and performance measures. The outcomes of these experiments shall help guide researchers in the choice of baseline algorithms to consider in their own research. In particular we found that one should consider algorithms of different types in any evaluation, as there appears to be no single method that is better than all others in all experimental configurations. Second, we ran these experiments with the help of a recent general evaluation framework for recommender systems [15], thus allowing other researchers to benchmark their new models within a defined environment and against already well-tuned baselines.

In terms of the outcomes of the experiments, our reproducibility study confirmed earlier findings that the latest models are not often the best performing ones, in particular for the modest-sized datasets that we considered in our evaluation. In our ongoing and future work, we plan to fine-tune our models also on larger datasets and to share these tuned models publicly. Thereby, we hope to reduce the often huge computational effort that other researchers would otherwise need to fine-tune all baseline models whenever they propose a new model. Over time, this collection of fine-tuned models for various datasets may represent a step towards a shared understanding of what represents the “state-of-the-art” in algorithms research. For these larger datasets, we also expect a more consistent and strong performance of deep learning models.

Besides accuracy metrics, our experiments included a number of beyond-accuracy metrics relating to popularity bias, novelty, fairness, and item coverage. Our results confirm earlier findings that there can be substantial differences between algorithms, e.g., in terms of their tendency to recommend popular items. Such algorithm tendencies can be of high relevance in practical application settings, e.g., when the goal is to support item discovery through the recommendations. An important observation in our research is that common metrics for novelty and fairness are tightly coupled and correlated with general popularity biases[13]. Future research might therefore strive to find alternative metrics that more often go beyond popularity as indicators for novelty, diversity, fairness, or serendipity.

In addition to this, a careful analysis on the effect of the optimization goals for hyperparameter tuning is missing in the literature. The results presented herein considered methods optimized for one specific accuracy-oriented metric, i.e., nDCG. But what would happen if other metrics are used for this optimization? It is true that there are strong correlations between some metrics, as discussed before, but it is also well-known that accuracy and beyond-accuracy measurements are typically inversely related, hence, the question of what "state-of-the-art" means in terms of these other metrics remains open and should be addressed in the future.

Finally, another aspect regarding the splitting strategy has to be taken into consideration. Here, we adopted a random hold-out splitting strategy with repeated experiments that became popular in recent literature. Together with k-folds cross-validation, they are representative of the evaluation protocols adopted in recent works. Nevertheless, random-based splitting strategies undoubtedly favor some methods since information regarding the future general users' behavior is exploited in the training phase. More realistic time-aware splitting strategies should be investigated to study how much they impact the overall ranking of recommendation systems.

4.4 Summary

In this chapter, a more rigorous investigation of the issue of repeatability in recommender systems was carried out. Our research went in two different directions: on the one hand, we demonstrated that it is possible to ensure the full reproducibility of the results obtained by stating the set-up and thanks in our study in a very careful manner. These results had previously been deemed significant by the community. On the other

[13] In theory, the Gini index is not necessarily tied to popularity biases, but with the typical long-tail distributions it usually captures a concentration of items in the "short head".

hand, state-of-the-art collaborative filtering family models, datasets that are commonly utilized in common experimental set-ups, and assessment techniques that are routinely applied were investigated in more depth. This attempt was undertaken in order to study the dominance of certain models over others in an assessment scenario common for this research field. The purpose of this investigation was to determine which models were more successful than others. In both of these insights, we added a piece that was missing from the puzzle of evaluating these models in the relevant scientific literature. Specifically, we added an in-depth analysis of these models with respect to novelty, diversity, and bias metrics. This was the piece that was missing from the puzzle. In light of these investigation presumptions, we demonstrated how the most recent models are not always the ones with the highest performance levels. In addition to this, there is not an in-depth study that examines the influence of the optimization objectives on the process of hyperparameter tuning anywhere in the published research. The findings that were provided here took into account approaches that were optimized for one particular accuracy-oriented parameter, which was nDCG. But what would occur if other measures were used for this process of optimization? It is true that there are strong correlations between some metrics, as was previously discussed, but it is also common knowledge that accuracy and beyond-accuracy measurements are typically inversely related. As a result, the question of what "state-of-the-art" means in terms of these other metrics remains open, and it is something that should be addressed in the future.

Chapter 5

On the trade-offbetween accuracy and beyond accuracy in modern recommender systems

RSs have shown to be extremely useful in understanding the preferences of users and finding the goods that are most interesting to them depending on their choices. Latent factor models have dominated the RSs landscape and contributed to their widespread adoption. With the advent of neural approaches (see Section 3), proposed new models have been tried to overcome the stumbling block of linearity inherent in those models. These models have been proposed in an effort to overcome this inherent linearity. We put graph-based methods among the most promising new approaches that have been developed. These approaches look at the information contained in RSs via the lens of graphs. When describing people and items as nodes with latent representations and their interaction as edges, the data may be naturally represented as a user-item bipartite graph. This can be accomplished by modeling the data as follows: The so-called message-passing schema successfully distills the collaborative signal by updating the initial photos of nodes by pooling contributions from both nearby and far-away neighbors [261].

In this chapter, we will evaluate the performance of graph-based RSs with a particular emphasis on beyond accuracy and the part that hyperparameters in these models play in determining how these dimensions are affected. We will examine how well various graph-based approaches can offer suggestions that are free from bias and concerns about fairness for various users. We will do this by revisiting the traditional methodology of forwarding messages and making an appropriate inquiry. In addition, we present a strategy that, by expanding on our recently developed formulation of the

message-passing process, can consistently suggest diverse and novel recommendations without sacrificing accuracy.

5.1 Introduction

Recommender systems (RSs) are ubiquitous and utilized in a wide range of domains from e-commerce and retail to media streaming and online advertising. Personalization, or the system's ability to suggest relevant and engaging products to users, has long served as a key indicator for gauging the success of RSs. In recent decades, collaborative filtering (CF) [83], the predominant modeling paradigm in RSs, has shifted from neighborhood techniques [83, 206, 221] to frameworks based on the learning of users' and items' latent factors [141, 201, 270]. The former methods directly suggest products to users based on similarities encoded in the historical interactions. The alternative representation learning approach represent users and items as continuous vectors (also known as embeddings) in a shared space, making them readily comparable. A notable example of the latter category is BPR-MF, which uses a rank-optimization framework based on personalized Bayesian ranking (BPR) to learn user and item embeddings [197]. More recently, deep learning (DL) models have been proposed to overcome the linearity of traditional latent factors approaches making them a popular tool for representation learning in academic research and industry. However, data sparsity and cold-start problems are common in such methods, because users often interact with a subset of the catalog's items.

Among these DL algorithms, graph-based methods view the data in RSs from the perspective of graphs. By modelling users and items as nodes with latent representations and their interaction as edges, the data can be naturally represented as a user-item bipartite graph. By iteratively aggregating contributions from near- and long-distance neighborhoods, the so-called message-passing schema updates nodes' initial representations thus effectively distilling the collaborative signal [261]. Early works [38, 280] adopt the vanilla graph convolutional network (GCN) [136] architecture, paving the way to advanced algorithms lightening its message-passing schema [58, 109] and exploring different settings of graph sampling strategies [267], while recent approaches propose simplified formulations [173, 192] that optionally transfer the graph CF paradigm to different spaces [226, 234]. As some graph edges may provide noisy contributions to the message-passing schema [252], a parallel research path learns meaningful user-item interactions [241, 260, 264]. In this context, explainability is the

natural next step [169] towards the disentanglement of user-item connections into a set of possible user intents [262, 268].

Graph CF techniques have historically centered on the enhancement of system accuracy. The various strategies have pushed the performance in terms of accuracy without focusing on the beyond-accuracy dimensions. Conversely, recent works [173, 226] highlight critical limitations in the adoption of graph convolution to explore users' and items' neighborhoods. Starting from the idea described in [109], they propose alternative reformulations of GCN for the recommendation task, providing simplified and lighter versions which go beyond the traditional concept of multi-hop message-passing. By comparing these latter approaches to the ones described earlier, we might categorize them all into two families, namely, graph recommendation techniques performing *explicit* (e.g., [58, 109, 261, 262]) and *implicit* (e.g., [173, 226]) message-passing.

On the other side, the adoption of DL approaches to the recommendation task has raised the issues connected to fairness of RSs. There is a rising awareness of the fairness of machine learning (ML) models in automated decision-making tasks, such as classification [45, 155] and ranking tasks [53, 56], with RSs acting as a prominent example of the latter. The concept of fairness in recommendation is multifaceted. Specifically, the two core aspects to categorize recommendation fairness may be summarized as (1) the primary parties engaged (consumers vs. producers) and (2) the type of benefit provided (exposure vs. relevance). Item suppliers are more concerned with exposure fairness than customers because they want their products to be more well-known and visible (**P**roducer fairness). However, from the customer's perspective, relevance fairness is of utmost importance, and hence system designers must ensure that exposure of items is equally effective across user groups (**C**onsumer fairness). According to a recent study, nine out of ten publications on recommendation fairness have concentrated on either C-fairness or P-fairness [181], disregarding their joint evaluation between C-fairness, P-fairness, and the system accuracy. Some recent works have addressed producer fairness [43, 170, 171, 235, 291, 293] and consumer fairness [88, 153, 194, 254, 258, 269] in RSs area. However, there is a notable *knowledge gap* in the literature about the effects of graph characteristics on the three objectives of C-fairness, P-fairness, and system accuracy.

This chapter aims to complement the previous research and provide answers to pending research problems in graph CF, such as, how different graph CF perform with respect to the three evaluation objectives. By measuring them in terms of *overall accuracy*, *user fairness*, and *item exposure*, we will look at these aspects in further

detail[1]. We seek to understand how and why the neighborhood exploration strategy and (optionally) depth may influence novelty and diversity recommendation metrics in graph collaborative filtering. To this aim, we run extensive experiments by training and evaluating six state-of-the-art graph CF models on three popular recommendation datasets.

5.2 Related Work

The approach proposed by Berg et al. [38] is the first attempt to address the recommendation task through a graph-based architecture. The authors implement a graph autoencoder that labels its edges with users' ratings to perform link prediction. Ying et al. [280] design a graph convolutional network for a web-scale recommendation to produce high-quality image recommendations for the Pinterest platform, efficiently exploiting random walk and item's multimodal side information. Wang et al. [261] present neural graph collaborative filtering (NGCF), whose propagation layer aggregates the messages from the neighborhood considering the similarity between each neighbor node and its ego node. While providing higher performance to previous state-of-the-art solutions, NGCF (and GCN more generally) show limitations later addressed by He et al. [109]. Their idea is to lighten GCN's traditional layer structure and reach superior accuracy performance by removing non-linearities and node embedding transformation in the propagation layer (LightGCN). The latest approaches try to take a step further to the LightGCN strategy by allowing theoretically unlimited propagation layers [173] and revisiting the concept of graph convolution for recommendation and node embedding smoothness under the lens of graph signal processing [226].

While aggregating messages from neighbor nodes into the ego node, not all received contributions have the same importance. The pioneering work by Velickovic et al. [252], called graph attention network (GAT), takes advantage of attention mechanisms to weigh the different influences of neighbor nodes on the ego node. Inspired by this rationale, several recent works in recommendation seek to assess the relative importance of interacted items on users who are involved in those interactions. In the last few years, recommendation tasks such as session-based recommendation [228, 274, 275] and sequential recommendation [271, 286] have been widely addressed by using attention mechanisms on graphs. Attention mechanisms may be also beneficial when the informative content conveyed by the bipartite user-item graph is enhanced

[1]In the rest of the chapter, when no confusion arises, we will refer to C-fairness with user fairness, to P-fairness with item exposure, and to their combination as CP-fairness.

by additional side information, like knowledge graphs [260], heterogeneous information networks [264], or multimodal items' content [241]. Exploiting attention to disentangle the aspects underlying node interactions may represent a fundamental step toward explainability [169]. Following this direction, the work by Wang et al. [262], named disentangled graph collaborative filtering (DGCF), and the method presented in Wu et al. [268], propose to disentangle user-item connections into a set of possible user intents.

The above-cited works leverage what we might define as an *explicit* message aggregation, meaning that it is always possible to derive a formulation where user and item node embeddings are *explicitly* updated through their multi-hop neighbors. Conversely, following a different rationale, more recent approaches take a step further and try to rethink the message-passing schema by allowing theoretically unlimited propagation hops [173] and revisiting the concept of graph convolution and node embedding smoothness through the lens of graph signal processing [226]. To distinguish such techniques from the *explicit* ones, in this work, we introduce the concept of *implicit* message-passing, where message aggregation is replaced and improved through ad-hoc mathematical proxies.

5.3 Graph Collaborative Filtering: Overview and Formal Taxonomy

5.3.1 Preliminaries

Let $\mathcal{U}$ be the set of N users, and $\mathcal{I}$ the set of M items in the system, respectively. We represent the observed interactions between users and items in a binary format (i.e., unary feedback). Specifically, let $\mathbf{R} \in \mathbb{R}^{N \times M}$ be the user-item feedback matrix, where $r_{u,i} = 1$ if user $u \in \mathcal{U}$ and item $i \in \mathcal{I}$ have a recorded interaction, $r_{u,i} = 0$ otherwise. The $\mathbf{R}$ matrix is used to build the adjacency matrix $\mathbf{A} \in \mathbb{R}^{(N+M)\times(N+M)}$ that encodes the bi-directional user-item connections but not their self-connections.

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{R} \\ \mathbf{R}^\top & \mathbf{0} \end{bmatrix} \tag{5.1}$$

Following the above preliminaries, we introduce $\mathcal{G} = (\mathcal{U}, \mathcal{I}, \mathbf{R})$ as the bipartite and undirected graph connecting users and items (the graph nodes) when there exists a recorded bi-directional interaction among them (the graph edges). Nodes features for

user $u \in \mathcal{U}$ and $i \in \mathcal{I}$ are suitably encoded as the embeddings $\mathbf{e}_u^{(0)} \in \mathbb{R}^d$ and $\mathbf{e}_i^{(0)} \in \mathbb{R}^d$, with $d << N, M$. As the item version is dual-derived, we report user-side formulas in the following sections.

5.3.2 Updating node representation through message-passing

We seek to update the initial representation of users' and items' nodes by leveraging the graph topology from $\mathcal{G}$. In this respect, the message-passing schema has recently gained attention in the literature. The algorithm works by aggregating the information (i.e., the *messages*) from the *neighbor* nodes into the *ego* node, and the process is recursively performed for multiple hops thus exploring wider neighborhood portions.

In general, the message-passing for l hops is:

$$\mathbf{e}_u^{(l)} = \omega\left(\left\{\mathbf{e}_{i'}^{(l-1)}, \forall i' \in \mathcal{N}(u)\right\}\right), \quad \mathbf{e}_i^{(l)} = \omega\left(\left\{\mathbf{e}_{u'}^{(l-1)}, \forall u' \in \mathcal{N}(i)\right\}\right). \tag{5.2}$$

where $\omega(\cdot)$ and $\mathcal{N}(\cdot)$ are the aggregation function and neighborhood node set, respectively. A reworking of Equation (5.2) for $l \in \{2,3\}$ would allow *same-* and *different-*type node interactions explicitly emerge:

$$\begin{cases} \textit{Same-type node representation} & \underbrace{\mathbf{e}_u^{(2)}}_{\text{(user)}} = \omega\Big(\Big\{\omega\Big(\Big\{\underbrace{\mathbf{e}_{u''}^{(0)}}_{\text{(user)}}, \forall u'' \in \mathcal{N}(i') \setminus \{u\}\Big\}\Big), \forall i' \in \mathcal{N}(u)\Big\}\Big) \\ \textit{Different-type node representation} & \underbrace{\mathbf{e}_u^{(3)}}_{\text{(user)}} = \omega\Big(\Big\{\omega\Big(\Big\{\omega\Big(\Big\{\underbrace{\mathbf{e}_{i'''}^{(0)}}_{\text{(item)}}, \forall i''' \in \mathcal{N}(u'') \setminus \{i''\}\Big\}\Big), \\ & \qquad \forall u'' \in \mathcal{N}(i') \setminus \{u''\}\Big\}\Big), \forall i' \in \mathcal{N}(u)\Big\}\Big). \end{cases} \tag{5.3}$$

To generalize Equation (5.3), ego node updates after an **even** and an **odd** number of explored hops adopt **same-** and **different-**type node connections, respectively (e.g., user-user/item-item and user-item/item-user as evident from the Equation (5.3)). Under such a reformulation, and differently from other works in the literature, we will count the number of explored hops as follows: $\mathbf{e}_*^{(2h)}, \forall h \in \{1,2,\ldots,H\}$ as h **same-**type node connections, and $\mathbf{e}_*^{(2h-1)}, \forall h \in \{1,2,\ldots,H\}$ as h **different-**type node connections. In the following, we use the general message-passing formulation to introduce the standard graph convolutional network (GCN), and then its recent CF applications.

The baseline: graph convolutional network (GCN). The standard graph convolutional network from [136] performs feature transformation, message aggregation, and non-linearity application through a one-layer neural network, the element-wise addition, and the ReLU activation function, respectively. Let us consider $\mathbf{W}^{(l)} \in \mathbb{R}^{d_{l-1} \times d_l}$

and $\mathbf{b}^{(l)} \in \mathbb{R}^{d_l}$ as the weight matrix and the bias for the l-th explored hop. The message-passing for user u is:

$$\mathbf{e}_u^{(l)} = \text{ReLU}\left(\sum_{i' \in \mathcal{N}(u)} \left(\mathbf{W}^{(l)}\mathbf{e}_{i'}^{(l-1)} + \mathbf{b}^{(l)}\right)\right). \tag{5.4}$$

GCN for collaborative filtering. Inspired by the vanilla GCN message-passing approach, the authors from [261] propose neural graph collaborative filtering (NGCF). At each hop exploration, the model aggregates the neighborhood information as well as the inter-dependencies among the ego and the neighborhood nodes. More formally, the aggregation could be formulated as follows:

$$\mathbf{e}_u^{(l)} = \text{LeakyReLU}\left(\sum_{i' \in \mathcal{N}(u)} \left(\mathbf{W}_{\text{neigh}}^{(l)}\mathbf{e}_{i'}^{(l-1)} + \mathbf{W}_{\text{inter}}^{(l)}\left(\mathbf{e}_{i'}^{(l-1)} \odot \mathbf{e}_u^{(l-1)}\right) + \mathbf{b}^{(l)}\right)\right). \tag{5.5}$$

where LeakyReLU is the activation function, $\mathbf{W}_{\text{neigh}}^{(l)} \in \mathbb{R}^{d_{l-1} \times d_l}$ and $\mathbf{W}_{\text{inter}}^{(l)} \in \mathbb{R}^{d_{l-1} \times d_l}$ are the neighborhood and inter-dependencies weight matrices, respectively, while $\odot$ is the hadamard product.

He et al. [109] propose a light convolutional network, namely LightGCN, with the rationale to simplify the message-passing schema from GCN and NGCF by dropping feature transformations (i.e., the weight matrices and biases) and the non-linearity applied after the message aggregation. Specifically, they implement:

$$\mathbf{e}_u^{(l)} = \sum_{i' \in \mathcal{N}(u)} \mathbf{e}_{i'}^{(l-1)}. \tag{5.6}$$

This variation shows superior accuracy performance to the state-of-the-art. A slightly different solution [58] is able to outperform LightGCN on accuracy level.

5.3.3 Weighting the importance of graph edges

The message-passing schema is inherently designed to aggregate into the ego node all messages coming from its neighborhood. Nevertheless, the *binary* nature of the user-item feedback (i.e., 0/1) would suggest that not all recorded user-item interactions necessarily hide the same importance to the nodes they involve.

In general, let $a_{y \to x}^{(l)}$ be the importance of the neighbor node y on its ego node x after l explored hops. We re-write the formulation of the message-passing after l explored hops (presented in Equation (5.2)) as:

$$\mathbf{e}_u^{(l)} = \omega\left(\left\{a_{i' \to u}^{(l)}\mathbf{e}_{i'}^{(l-1)}, \forall i' \in \mathcal{N}(u)\right\}\right). \tag{5.7}$$

The baseline: graph attention network (GAT). Attention mechanisms weighting the contribution of neighbor messages before aggregation have reached considerable success in the GCN-related literature. The original study [252] proposes the following message-passing formulation:

$$\begin{aligned}\mathbf{e}_u^{(l)} &= \sum_{i' \in \mathcal{N}(u)} \left(a_{i' \rightarrow u}^{(l)} \mathbf{W}_{\text{neigh}}^{(l)} \mathbf{e}_{i'}^{(l-1)} + \mathbf{b}^{(l)} \right) \\ &= \sum_{i' \in \mathcal{N}(u)} \left(\alpha \left(\mathbf{e}_{i'}^{(l-1)}, \mathbf{e}_u^{(l-1)} \right) \mathbf{W}_{\text{neigh}}^{(l)} \mathbf{e}_{i'}^{(l-1)} + \mathbf{b}^{(l)} \right).\end{aligned} \tag{5.8}$$

where $a_{i' \rightarrow u}^{(l)} = \alpha \left(\mathbf{e}_{i'}^{(l-1)}, \mathbf{e}_u^{(l-1)} \right)$ is the importance function depending on the lastly-calculated embeddings of the neighbor and the ego nodes.

GAT for collaborative filtering. The authors from [262] design a message-passing schema which calculates the importance of neighborhood nodes on their ego nodes by disentangling the various intents underlying each user-item interaction. Specifically, similarly to [109] and [58], they propose the following embedding update formulation:

$$\begin{aligned}\mathbf{e}_u^{(l)} &= \sum_{i' \in \mathcal{N}(u)} a_{i' \rightarrow u}^{(l)} \mathbf{e}_{i'}^{(l-1)} \\ &= \sum_{i' \in \mathcal{N}(u)} \alpha \left(\mathbf{e}_{i'}^{(l-1)}, \mathbf{e}_u^{(l-1)}, K, T \right) \mathbf{e}_{i'}^{(l-1)}.\end{aligned} \tag{5.9}$$

where $a_{i' \rightarrow u}^{(l)} = \alpha \left(\mathbf{e}_{i'}^{(l-1)}, \mathbf{e}_u^{(l-1)}, K, T \right)$ is the importance function depending on the lastly-calculated embeddings of the neighbor and the ego nodes, the total number of intents K, and the total number of routing iterations T to repeat the disentangling procedure.

5.3.4 Reformulating Explicit message-passing

Starting from the novel model classification for graph collaborative filtering outlined in this chapter (i.e., neighborhood exploration approaches leveraging *explicit* or *implicit* message-passing), in this section we propose a simple (but useful) reformulation for the former family where *same*- and *different*-type node interactions (e.g., user-user and user-item, respectively) are formally highlighted.

The two-hop node update in Equation (5.3) is further expanded through the one-hop node update in Equation (5.2):

$$\begin{aligned}\mathbf{e}_u^{(2)} &= \omega\Big(\Big\{\omega\Big(\Big\{\mathbf{e}_{u''}^{(0)}, \underbrace{\forall u'' \in \mathcal{N}(i') \setminus \{u\}}_{\text{2-hop}}\Big\}\Big), \underbrace{\forall i' \in \mathcal{N}(u)}_{\text{1-hop}}\Big\}\Big) \\ \mathbf{e}_i^{(2)} &= \omega\Big(\Big\{\omega\Big(\Big\{\mathbf{e}_{i''}^{(0)}, \underbrace{\forall i'' \in \mathcal{N}(u') \setminus \{i\}}_{\text{2-hop}}\Big\}\Big), \underbrace{\forall u' \in \mathcal{N}(i)}_{\text{1-hop}}\Big\}\Big)\end{aligned} \tag{5.10}$$

where set differences are used to avoid node duplicates. After two hops, the node embeddings of user u and item i get the contributions of those users u'' and items i'' for whom there exists a *user-item-user* path connecting u with u'', and an *item-user-item* path connecting i with i'', respectively (Figure 5.1a). Such paths link *same*-type nodes. In a similar manner, let us apply the general formula from Equation (5.2) to the three-hop node update:

$$\mathbf{e}_u^{(3)} = \omega\left(\left\{\mathbf{e}_{i'}^{(2)}, \forall i' \in \mathcal{N}(u)\right\}\right), \quad \mathbf{e}_i^{(3)} = \omega\left(\left\{\mathbf{e}_{u'}^{(2)}, \forall u' \in \mathcal{N}(i)\right\}\right) \tag{5.11}$$

which we expand through Equation (5.10):

$$\begin{aligned}\mathbf{e}_u^{(3)} &= \omega\Big(\Big\{\omega\Big(\Big\{\omega\Big(\Big\{\mathbf{e}_{i'''}^{(0)}, \underbrace{\forall i''' \in \mathcal{N}(u'') \setminus \{i''\}}_{\text{3-hop}}\Big\}\Big), \\ &\quad \underbrace{\forall u'' \in \mathcal{N}(i') \setminus \{u''\}}_{\text{2-hop}}\Big\}\Big), \underbrace{\forall i' \in \mathcal{N}(u)}_{\text{1-hop}}\Big\}\Big) \\ \mathbf{e}_i^{(3)} &= \omega\Big(\Big\{\omega\Big(\Big\{\omega\Big(\Big\{\mathbf{e}_{u'''}^{(0)}, \underbrace{\forall u''' \in \mathcal{N}(i'') \setminus \{u''\}}_{\text{3-hop}}\Big\}\Big), \\ &\quad \underbrace{\forall i'' \in \mathcal{N}(u') \setminus \{i''\}}_{\text{2-hop}}\Big\}\Big), \underbrace{\forall u' \in \mathcal{N}(i)}_{\text{1-hop}}\Big\}\Big)\end{aligned} \tag{5.12}$$

After three hops, the node embeddings of user u and item i get the contributions of those items i''' and users u''' for whom there exists a *user-item-user-item* path connecting u with i''', and an *item-user-item-user* path connecting i with u''', respectively (Figure 5.1b). In this case, such paths link *different*-type nodes.

This reformulation outlines two neighborhood exploration types, propagating messages through *same*- and *different*-type nodes after an even and an odd number of hops, respectively. While previous works assess recommendation performance when

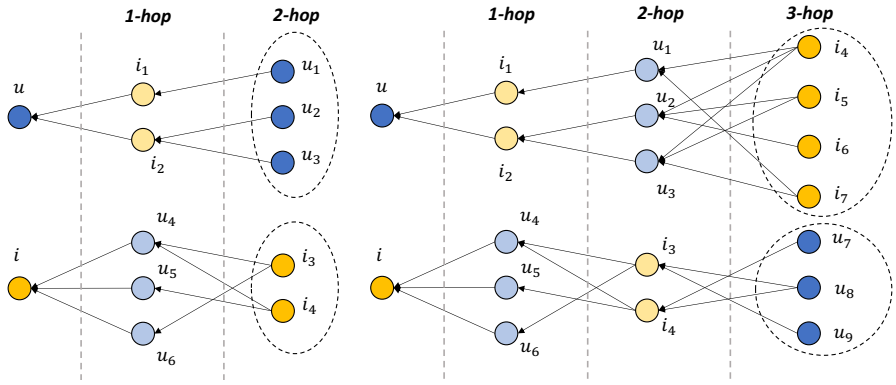


Fig. 5.1 User and item neighborhood exploration after (a) 2 and (b) 3 hops. Contributions to the ego node update are highlighted through dashed ovals. Edge direction indicates the message propagation from neighbor to ego nodes.

indistinctly increasing the hop numbers, we provide a finer evaluation based on the type of the explored nodes. In the next sections, we will count hops following the introduced categorization. For example, *same*-type node explorations after 1 and 2 hops refer to the paths *user*-*item*-*user* and *user*-*item*-*user*-*item*-*user*, respectively, while *different*-type node explorations after 1 and 2 hops refer to the paths *user*-*item* and *user*-*item*-*user*-*item*, respectively.

5.3.5 Going beyond message-passing

The recent graph learning literature [55, 295] has outlined the phenomenon of *over-smoothing*, that leads node representations to become more similar as more hops are explored. The issue is generally tackled by limiting the neighborhood exploration to (maximum) three hops, and to two hops when attention mechanisms are introduced. Noteworthy, the idea of improving accuracy by restricting the number of explored neighborhoods conflicts with the rationale behind collaborative filtering and is counter-intuitive. This awareness led works such as [173] and [226] to surpass and simplify the traditional concept of message-passing. UltraGCN [173] adopts negative sampling to contrast over-smoothing and additional objective terms to (i) approximate the infinite neighborhood exploration and (ii) mine relevant "unexpected" node-node interactions such as the item-item ones. Conversely, GFCF [226] translates the graph-based recommendation task into the graph signal processing domain to obtain a closed-form formulation.

5.3.6 A taxonomy of graph CF approaches

The model formalization from the previous sections allows to design a general pipeline of recommendation stages in graph collaborative filtering (Figure 5.2): 1) the input data, represented in the form of the user-item bipartite and undirected graph and the adjacency matrix; 2) users' and items' nodes are associated with an initial latent representation (embeddings); 3) such nodes representations are iteratively updated through message-passing leveraging *same-* and *different*-type node connections, and the schema could involve the weighting of the neighborhood before message aggregation; 4) after the node representation refining, node latent representations are used to predict the final user-item score, and build the top-k list of items for each user. It is worth underlining that the node latent representation updating stage (i.e., *explicit* message-passing) may be replaced (or complemented) via other mathematical re-formulations and proxies of the usual schema (i.e., *implicit* message-passing).

The outlined recommendation stages in graph collaborative filtering suggest a **taxonomy** of approaches. Specifically, we recognize two main dimensions adopted during the design of a graph-based recommendation system, namely:

- **Nodes representation:** indicates the latent representation strategy to model users' and items' nodes. It involves the *dimensionality* of node embeddings, and the possibility of *weighting* the contribution of each neighbor node before the aggregation into the ego node.
- **Neighborhood exploration:** refers to the procedure to explore the multi-hop neighborhoods of the ego node during latent representation updating of nodes. It involves the type of *node-node connections* which are explored, and the *message-passing* schema (i.e., explicit or implicit as previously defined).

In the following, we will leverage the introduced taxonomy to provide a novel evaluation on state-of-the-art models in graph collaborative filtering.

5.4 Motivating example and Trade-offAnalysis

An in-depth comparison of eight leading graph CF models is carried out to provide context for our study. The graph-based models selected include: GCN-CF [136], GAT-CF [252], NGCF [261], LightGCN [109], DGCF [262], LR-GCCF [58], UltraGCN [173], and GFCF [226], which are tested against four classical CF baselines, namely, UserKNN, ItemKNN, BPRMF, and RP$^3\beta$ [190], on the Baby, Boys & Girls, and Men datasets

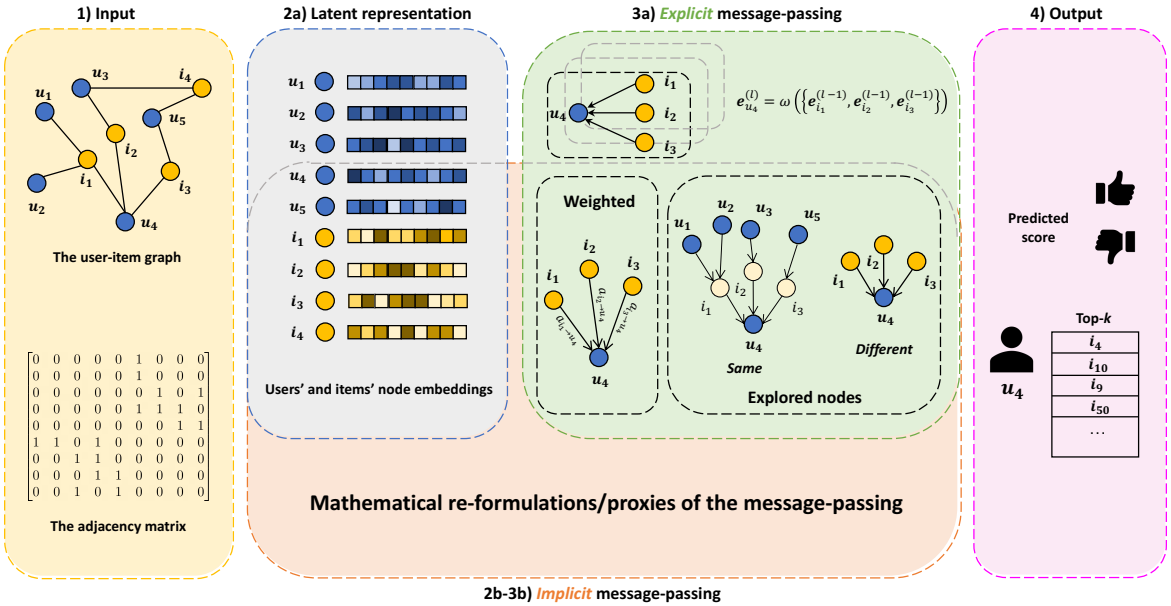


Fig. 5.2 Recommendation stages in graph CF. Starting from the user-item graph and the adjacency matrix (1), a latent representation of the users' and items' nodes is initialized (2a). Afterwards, the node embeddings are iteratively updated by exploring *same-* or *different-*type node-node connections through an ***explicit*** message-passing schema, which might be optionally weighted (3a). The final outcome is the predicted user-item score, which allows to build a ranked list of items (4). Note that the latent representation and explicit message-passing stages may be substituted (or complemented) through mathematical proxies (i.e., ***implicit*** message-passing).

from the Amazon catalog [182]. We train each baseline using a total of 48 unique hyper-parameter settings and then select the optimal configuration for each baseline that achieves the highest accuracy based on the validation set's results (following the original papers). Three evaluation objectives are analyzed: overall accuracy, user fairness, and item exposure. The values are scaled between 0 and 1 using min-max normalization. Then, for representation purposes, we exploit the 1's complement of *UMADrank* to adhere to the "higher is better" semantics. As a result, in each of the three dimensions, the values lay between $[0,1]$ with higher values indicating the better. Please note that this experimental evaluation is not the main focus of this work but it is the starting point for the analysis we present here.

5.4.1 Reproducibility and Evaluation Protocol

As a pre-processing stage, we randomly sample 60k interactions and drop users and items with less than five interactions to avoid the cold-start effect [106, 107]. The dataset statistics are: (1) Baby has 5,842 users, 7,925 items, 35,475 interactions; (2)

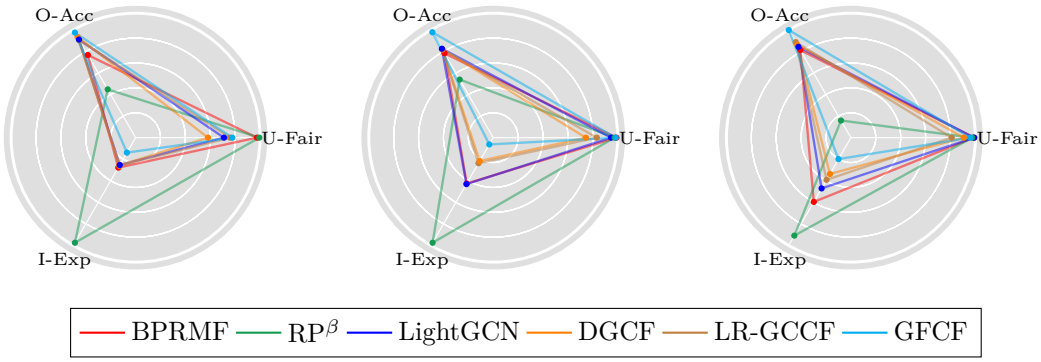


Fig. 5.3 Kiviat diagrams indicating the different performance of selected pure and graph CF recommender models on overall accuracy (i.e., O-Acc, calculated with the *nDCG*@20), item exposure (i.e., I-Exp, calculated with the *APLT*@20), and user fairness (U-Fair, calculated with the *UMADrat*@20). For all dimension the higher values means the better in that dimension.

Boys & Girls has 3,042 users, 12,912 items, 35,762 interactions; (3) Men has 3,909 users, 27,656 items, 51,519 interactions. Datasets are split using 70-10-20 train/validation/test hold-out strategy. Baselines are trained through grid search, with a batch size of 256 and a number of epochs of 400.

As for the *overall accuracy*, we use the recall (*Recall@k*) and the normalized discounted cumulative gain (*nDCG@k*). Concerning the *item exposure*, we focus on: (1) item novelty [250, 251] through the expected free discovery (*EFD@k*) measuring the expected number of recommended known items which are also relevant; (2) item diversity [225] with the 1's complement of the Gini index (*Gini@k*) which estimates how unequally a recommender system suggests heterogeneous items to users; (3) the average percentage of items from the long-tail (*APLT@k*) which are recommended in users' lists [5] to calculate how recommendation is biased towards popular items. *User fairness* indicates how equally each user group receives accurate recommendations. Users are split into quartiles based on their engagement level –i.e., the number of items each user interacted with. We then measure *UMADrat@k* [71] and the *UMADrank@k* [71], where the former stands for the average deviation in the predicted ratings among users groups, while the latter represents the average deviation in the recommendation accuracy (calculated in terms of *nDCG@k*) among users groups. The best hyper-parameter configurations are found by considering *Recall@20* on the validation.

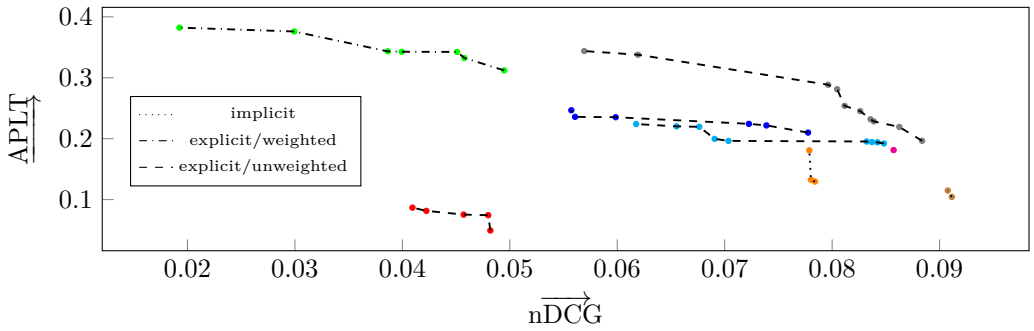
According to the results presented in Figure 5.3, we observe that graph CF models are significantly more accurate than pure CF models, even if the latter perform far better in terms of item exposure. Moreover, there is no clear winner on the user fairness dimension: pure CF models show promising performance while some graph CF

models do not achieve remarkable results. It is evident that there is a trade-offbetween the three evaluation goals. This encourages research to **tailor** our investigation to the cause in terms of the **different strategy patterns** recognized in the graph CF literature.

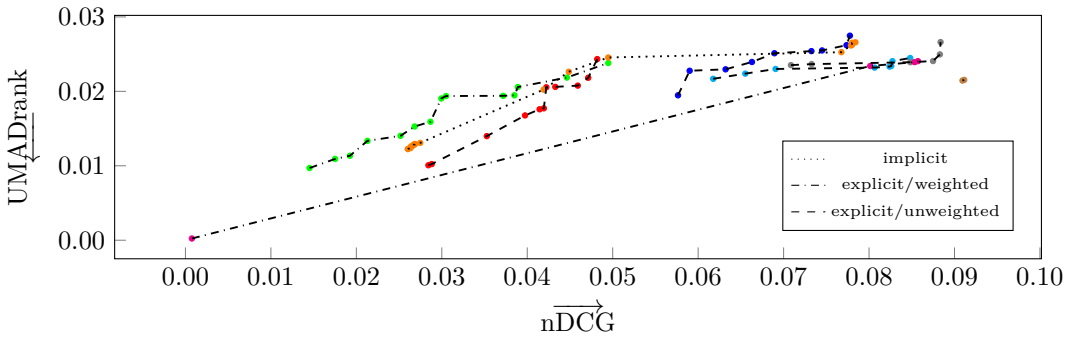
5.4.2 Trade-offAnalysis

We study how the graph CF baselines can balance the trade-offamong overall accuracy, item exposure, and user fairness. Specifically, we oppose *nDCG/APLT* and *nDCG/UMADrank* for assessing the overall accuracy/item exposure and overall accuracy/user fairness trade-offs, respectively. Indeed, the Pearson correlation (PC) for Amazon Men (i.e., -0.042, and -0.392, respectively) suggests that searching for a trade-offis feasible. Moreover, we evaluate the trade-offbetween the two fairness sides (i.e., consumer and producer) by contrasting *APLT* and *UMADrank* (i.e., -0.177 PC). We select the **message-passing** and **weighting** of graph edges. Overall, we identify: (1) models with implicit message-passing (*implicit*); (2) models with explicit message-passing and neighborhood weighting (*explicit/weighted*); (3) models with explicit message-passing without neighborhood weighting (*explicit/unweighted*). For each pair of objectives, we consider the Pareto optimal solutions of the baselines laying on the model-specific Pareto frontier – a solution is Pareto optimal if no other solution can improve an objective without hurting the other one. Figure 5.4 plots graph models Pareto frontiers in the common objective function spaces related to the considered dimensions for the Amazon Men dataset. Please notice the different axis' scales across the graphics. The colors of Pareto optimal configurations refer to each model, while the lines style is used to distinguish the categories: dotted lines for *implicit*, dash dot lines for *explicit/weighted*, and dashed lines for *explicit/unweighted*.

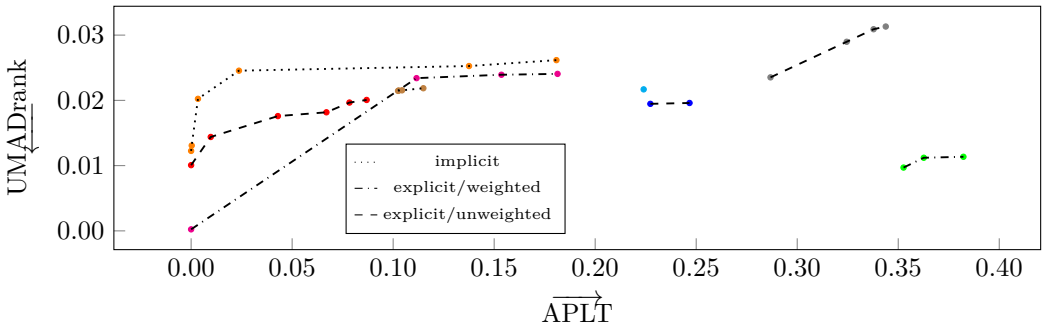
- **Accuracy/Item Exposure.** By considering Figure 5.4a, we observe that the *explicit/weighted* models exhibit a trade-off, as they maximize either *nDCG* (i.e., DGCF) or *APLT* (i.e., GAT-CF), but not both. This is expected since DGCF is designed to improve GAT-CF precision. Unfortunately, DGCF succeeded in it at the expense of item exposure. In contrast to these models, *explicit/unweighted* baselines can accommodate the trade-offbecause they do not prioritize accuracy or item exposure exclusively. On the one hand, LR-GCCF provides the best performance in terms of *nDCG* and *APLT* simultaneously. Visually, this observation suggests LR-GCCF's Pareto frontier dominates those of the other *explicit/unweighted* models. GCN-CF, on the other hand, is the worst in relation to the trade-offbecause it



(a) Overall Accuracy/Item Exposure



(b) Overall Accuracy/User Fairness



(c) tem Exposure/User Fairness

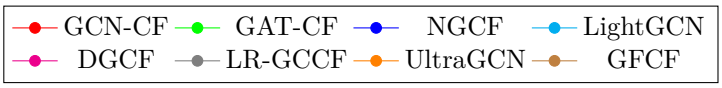


Fig. 5.4 Overall Accuracy/Item Exposure, Overall Accuracy/User Fairness, and Item Exposure/User Fairness trade-offs on Amazon Men, assessed through *nDCG/APLT*, *nDCG/UMADrank*, and *APLT/UMADrank*, respectively. Each point depicts a model hyper-parameter configuration set belonging to the corresponding Pareto frontier. Colors refer to a particular baseline, while lines styles discern their technical characteristics based on the proposed taxonomy. Arrows indicates the optimization direction for each metric on x and y axes.

is neither ideal for *nDCG* nor *APLT*. As for the *implicit* models, they appear to prioritize precision over the provision of long-tail items. Under this lens, the latest approaches seem to work towards increasing recommendation accuracy, even if at the detriment of the niche items exposure.

- **Accuracy/User Fairness.** Regarding the investigated trade-off, both measures incorporate the end-user perspective. Using Figure 5.4b as a reference, we observe that all baselines perform well from the *UMADrank* perspective. However, their recommendation accuracy performance varies. When dealing with this trade-off, there is no emerging graph CF strategy. This indicates that solutions working towards the improving of recommendation accuracy may be preferred by systems designers.

- **Item Exposure/User Fairness.** Here we assess to what extent graph CF models can treat final users fairly and recommend items from the long-tail. Similarly to above, Figure 5.4c shows the efficiency of all baselines from the users' viewpoint. The same observation does not hold for *APLT*. In this regard, it is possible to identify two sets of baselines: on the one hand, models that show poor performance in terms of item exposure (UltraGCN, DGCF, GCN-CF, and GFCF) and, on the other hand, models that exhibit an acceptable exposure for long-tail items (LightGCN, NGCF, LR-GCCF, and GAT-CF). Noteworthy, as for the latter group, NGCF, LR-GCCF, and LightGCN are *explicit/unweighted* models. In addition, the remaining model (i.e., GAT-CF) achieves better *APLT* values at the detriment of accuracy (as already outlined for the accuracy/item exposure trade-off). Furthermore, the same approach definitely provides the optimal solution for this trade-off. The leftover set of models appears to perform below the 50% of the *APLT* value reached by the best model on item exposure. Among them, the *implicit* baselines evidently confirm their trend to suggest popular items even more, as already noticed in the first part of this discussion. To summarize, since all models perform well on UMADrank, also in this case a system designer could be more interested in promoting models guaranteeing the best value for APLT. However, explicit/unweighted technical characteristics of the models can generally ensure a satisfactory trade-offbetween user fairness and item exposure.

5.5 The role of Neighborhood Exploration

Literature has widely shown the recommendation accuracy boost of such models to traditional (i.e., non-graph) CF baselines, their ability to produce *novel* and *diverse* recommendation lists [250, 251] remains poorly investigated. While the topic of *multi-objective* recommendation has been addressed only recently by few works in graph CF [235, 293], modern recommender systems are more and more required to reach a sufficient trade-offbetween accurate and novel/diverse recommendations [149, 229, 278], as a renewed need from both user's and business's perspectives [3, 7, 148].

Our contributions are threefold: (i) to the best of our knowledge, no previous work has evaluated approaches from the two recognized graph recommendation families (i.e., *explicit* and *implicit* message-passing) on a grid of accuracy/novelty/diversity recommendation metrics, (ii) to provide a fair comparison, we train all *explicit* message-passing models exploring the whole hop range 1-4, which also allows examining the accuracy/novelty/diversity trade-offon the neighborhood size, and (iii) we propose a simple reformulation of the *explicit* message-passing schema where *same*-type node connections (e.g., user-user) and *different*-type node connections (e.g., user-item) are formally highlighted, in an effort to unveil their influence on the metrics' trade-off.

In the following, we describe datasets, baselines, reproducibility details, evaluation protocol, and results of our work.

5.5.1 Reproducibility and Evaluation Protocol

Datasets. We adopt Movielens-1M [105], Amazon Digital Music [176], and Epinions [211]. Following a similar approach to [19], these datasets are binarized by retaining interactions with a score greater than 3 (Epinions already has an implicit version) and filtered through the p-core to avoid the cold-start effect [106, 107] which is out of the scope of this paper. Movielens-1M counts 5,915 users, 2,753 items, and 570,622 interactions, Amazon Digital Music counts 8,328 users, 6,275 items, and 99,400 interactions, and Epinions counts 14,341 users, 13,145 items, and 269,170 interactions. All datasets statistics are fully reported in Table 4.4.

Baselines. We evaluate graph recommendation models adopting *explicit* and *implicit* message propagation. Among the former we consider NGCF [261], LightGCN [109], DGCF [262], and LR-GCCF [58]. Regarding *implicit* message-passing we consider UltraGCN [173] and GFCF [226]

Reproducibility. Datasets are split into train/validation/test with the 80/10/10 hold-out. Models are trained by searching the best hyperparameters as in [39] and

setting search spaces according to the original works while fixing the number of epochs to 400 and batch size to 1024. Our implementation is based upon the Elliot framework for reproducible recommender systems [15]. To foster the future reproduction of our work, datasets, codes, and configuration files are made accessible to a public GitHub repository[2].

Evaluation. First, we use the recall (*Recall*@k) and the normalized discounted cumulative gain ($nDCG@k$) to measure the recommendation **Accuracy** of the baselines. Then, following [250, 251], we select the expected popularity complement ($EPC@k$) and the expected free discovery ($EFD@k$) as **Novelty** metrics [251], along with the 1's complement of the Gini index ($Gini@k$) and the Shannon entropy ($SE@k$) as **Diversity** metrics [225]. Both the $EPC@k$ and the $EFD@k$ account for long-tail items and measure the expected number of recommended unknown and known items, which are also relevant, respectively. The $Gini@k$ and the $SE@k$ calculate how unequally a recommender system shows different items to users. We set the *Recall*@20 as validation metric to follow the original papers. For each recommendation metric, higher values stand for better performance.

5.5.2 Results and Discussion

This section shows the recommendation performance of the tested baselines from a general and a finer evaluation of the accuracy/novelty/diversity trade-offs. All reported results refer to the top-20 recommendation lists.

Overall Performance. Table 5.1 depicts recommendation performance on accuracy, novelty, and diversity, when comparing *explicit* to *implicit* message-passing graph approaches in their best configuration.

Coherently with the literature, DGCF and LR-GCCF are steadily the best or the second-to-best models on accuracy (e.g., DGCF reaches the second-to-best *Recall* on Amazon Digital Music, while LR-GCCF obtains the best $nDCG$ on Movielens-1M). Approaches with *implicit* message aggregation (i.e., UltraGCN and GFCF) still compete with the other baselines on accuracy (e.g., GFCF is the best model on Amazon Digital Music for the *Recall* and the $nDCG$, and UltraGCN is the best technique on Epinions for the $nDCG$).

As for the accuracy/novelty/diversity trade-off, we see that, independently of the adoption of message-passing, accurate approaches can also produce novel recommendations (e.g., LR-GCCF and DGCF are the best and second-to-best approaches for

[2]https://github.com/sisinflab/Novelty-Diversity-Graph.

Table 5.1 Overall recommendation performance on accuracy, novelty, and diversity metrics for top-20 recommendation lists, when comparing **`explicit`** to **`implicit`** message propagation. Bold and underline stand for best and second-to-best values, respectively.

Models	Movielens-1M						Amazon Digital Music						Epinions					
	Accuracy		Novelty		Diversity		Accuracy		Novelty		Diversity		Accuracy		Novelty		Diversity	
	Recall	*nDCG*	*EPC*	*EFD*	*Gini*	*SE*	*Recall*	*nDCG*	*EPC*	*EFD*	*Gini*	*SE*	*Recall*	*nDCG*	*EPC*	*EFD*	*Gini*	*SE*
MostPop	0.1380	0.1099	0.0473	0.5365	0.0105	5.2156	0.0319	0.0154	0.0029	0.0263	0.0031	4.3832	0.0467	0.0224	0.0054	0.0489	0.0015	4.4358
Random	0.0077	0.0060	0.0036	0.0414	0.9105	11.4085	0.0017	0.0007	0.0002	0.0021	0.8929	12.5890	0.0015	0.0006	0.0002	0.0024	0.8789	13.6486
`Explicit` message-passing																		
NGCF	0.2535	0.1985	0.0929	1.0214	**0.1479**	8.9930	0.1127	0.0606	0.0109	0.1270	**0.4130**	**11.6953**	0.0792	0.0394	0.0096	0.1079	**0.2107**	**11.6255**
LightGCN	0.2712	0.2167	0.1013	1.1129	0.1465	9.0079	0.1189	0.0628	0.0113	0.1310	0.3148	11.2940	0.0914	0.0466	0.0115	0.1217	0.0759	9.7898
DGCF	0.2791	0.2231	0.1047	1.1490	0.1462	**9.0111**	0.1264	0.0674	0.0123	0.1400	0.2483	10.8904	**0.1046**	0.0536	**0.0132**	**0.1407**	0.0599	9.6502
LR-GCCF	**0.2876**	**0.2274**	**0.1056**	**1.1589**	0.1245	8.7438	0.1246	0.0664	0.0119	0.1388	0.4037	11.6542	0.0990	0.0504	0.0124	0.1377	0.1367	10.8977
`Implicit` message-passing																		
UltraGCN	0.2540	0.2045	0.0901	0.9921	0.0766	8.0334	0.1256	0.0675	0.0123	0.1382	0.1737	10.0458	0.1041	**0.0541**	0.0131	0.1397	0.0586	9.0948
GFCF	0.1685	0.1398	0.0583	0.6577	0.0117	5.4064	**0.1287**	**0.0744**	**0.0137**	**0.1544**	0.2392	10.4923	0.0946	0.0496	0.0115	0.1158	0.0277	7.5926

accuracy and novelty on Movielens-1M, and GFCF and UltraGCN provide superior accuracy performance on Amazon Digital Music and Epinions, respectively, with GFCF outperforming all other baselines on novelty, and UltraGCN getting slightly lower EPC and EFD values than DGCF). Unexpectedly, NGCF settles as the approach producing the most diverse lists of recommended items on all datasets (i.e., see *Gini* and SE) but cannot cope with the other baselines in terms of *Recall* and $nDCG$ (similarly to Random). Other graph models with *explicit* message-passing (especially DGCF and LR-GCCF) are placed in the best accuracy/diversity trade-offspot, as they are often the second-to-best approaches on diversity, with limited observable drops in the accuracy. Contrarily, techniques with *implicit* message aggregation always show the lowest diversity.

Observation 1. While the accuracy/novelty trade-offdoes not depend on the explicit/implicit message-passing, the accuracy/diversity trade-offis preserved only when explicitly propagating messages, at the expense of (limited) recommendation accuracy drops.

A finer trade-offs evaluation. Figure 5.5 shows the accuracy/novelty/diversity trade-offon Amazon Digital Music by varying the message-passing strategy (i.e., *explicit* and *implicit*) and neighbor exploration depth only for the former case. Specifically, we use the reformulation from Section 5.3.4 to separate explicit message propagation results into **`same`**- and **`different`**-type node explorations at 1/2 hops.

We confirm that, while UltraGCN and GFCF can compete well on the accuracy/novelty trade-offwith the other baselines (whatever the explored number of hops and node type), the opposite occurs on the accuracy/diversity trade-off. Indeed, higher accuracy values for UltraGCN and GFCF are obtained at the expense of significant

drops in their diversity, even compared to message propagation at 1 hop (e.g., DGCF surpasses them on diversity at the expense of a slightly lower accuracy in the **same**-node setting).

As for the influence of **same**- and **different**-type node explorations, wider explorations of the neighborhood almost always lead to improved accuracy/novelty and accuracy/diversity performance, independently of the explored node types (apart from the **same**-type settings for NGCF on the *Recall* and LR-GCCF on the *Recall* and the *EPC*). Noticeably, the exploration of 1 hop in the **same**-type node setting leads to a better trade-offin accuracy/novelty/diversity than the exploration of 2 hops in the **different**-node setting (e.g., LightGCN increases the *Recall* and the *EPC* without a significant variation of *Gini*, and DGCF slightly decreases the *Recall* and the *EPC*, but improves *Gini*).

Observation 2. To confirm observation 1, explicit message propagation (even at 1 hop) can reach a better accuracy/diversity trade-offthan implicit propagation; then, same-type node explorations may lead to improved accuracy/novelty and accuracy/diversity trade-offs.

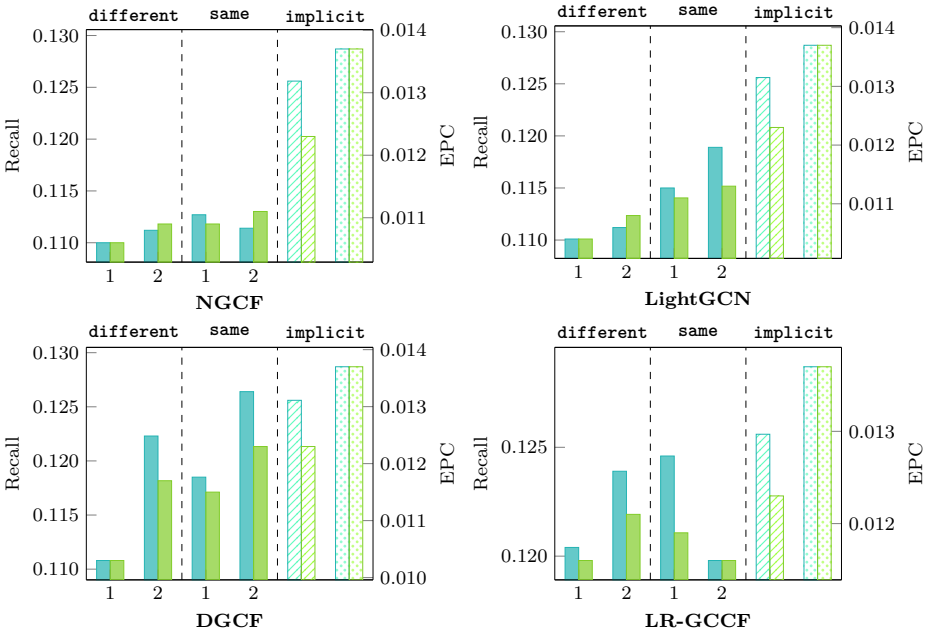
5.6 The showcase: Edge Graph Collaborative Filtering and Customer Reviews

In this section, we firt recall the causes of node error representation in graph collaborative filtering (see section 5.3.4) and demonstrate how existing neighborhood weighting procedures (e.g., attention mechanisms) may alleviate the issue at the expense of limited hop exploration. Second, we correct the representation error through an additional graph network where we enrich graph edge embeddings through opinion-aware review embeddings to smooth each neigh- bor node's importance on its ego node. We call our solution Edge Graph Collaborative Filtering (EGCF). Extensive experiments on three e-commerce datasets show that EGCF competes successfully with traditional, graph- and review-based approaches on accuracy and beyond-accuracy objectives, while a study on the number of ex- plored hops justifies the adopted configuration for EGCF.

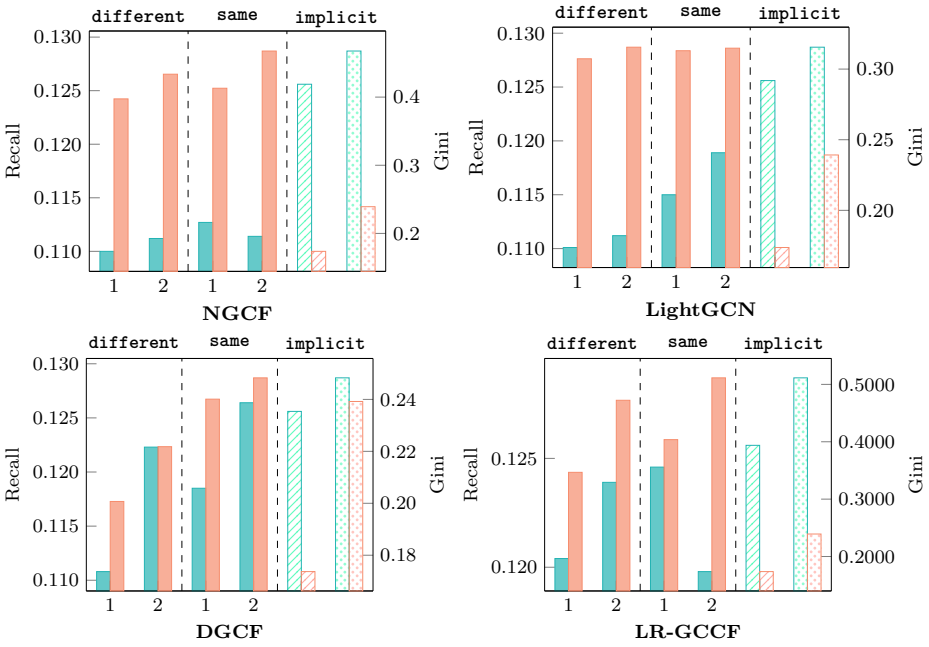
5.6.1 A limitation in the message-passing schema

The user formulation in Equation (5.2) can be expanded as follow:

$$\mathbf{e}_u^{(2)} = \omega\left(\left\{\omega\left(\left\{\mathbf{e}_{u'}, \forall u' \in \mathcal{N}(i) \setminus \{u\}\right\}\right), \forall i \in \mathcal{N}(u)\right\}\right) \tag{5.13}$$



(a) Accuracy — Novelty



(b) Accuracy — Diversity

Fig. 5.5 Accuracy/Novelty (a) and Accuracy/Diversity (b) trade-offs of graph models with **`explicit`** (i.e., filled bar plots) and **`implicit`** message-passing (i.e., patterned bar plots) on Amazon Digital Music for top-20 recommendation lists. As for explicit message-passing, results are further categorized into **`different`**- and **`same`**-node type explorations (i.e., the leftmost and central tabs in each plot, respectively), when varying the number of hops from 1 to 2. Accuracy, novelty, and diversity are assessed through *Recall* (in teal blue), *EPC* (in lime green), and *Gini* (in melon), respectively. Best viewed in color.

What emerges from the above formula is that, by propagating messages at two hops, the node embedding of user u is eventually refined through the contributions from other users who interacted with the same items as u. In other words, after two hops, *each user profile is influenced by the profiles of other users who rated the same items.*

Indeed, this assumption is aligned with the rationale behind collaborative filtering, i.e., similar users are likely to interact with the same items. However, not all user-item interactions (i.e., graph edges) may be equally important to the users and items involved. Thus, indiscriminately aggregating neighbor node embeddings into the ego node could, after multiple hops, harm the node updating process by bringing all contributions from the neighborhood, even the noisy ones. We interpret this as a *node representation error*, propagating with the exploration hops in the graph.

For this reason, contributions coming from each neighbor node are usually weighted before aggregating them into the ego nodes, modifying the presented formula as follows:

$$\mathbf{e}_u^{(2)} = \omega\left(\alpha_{i\to u}^{(2)}\left\{\omega\left(\left\{\alpha_{u'\to i}^{(1)}\mathbf{e}_{u'}, \forall u' \in \mathcal{N}(i)\setminus\{u\}\right\}\right), \forall i \in \mathcal{N}(u)\right\}\right) \tag{5.14}$$

where $\alpha_{j\to k}^{(l)}$ stands for the importance that the neighbor node j has on the ego node k after l hops. These weights are generally calculated by means of attention mechanisms, and depend on the embeddings of the neighbor and the ego nodes they refer to, e.g., $\alpha_{j\to k}^{(l)} = \varphi\left(\mathbf{e}_j^{(l-1)}, \mathbf{e}_k^{(l-1)}\right)$, where $\varphi(\cdot,\cdot)$ is, for example, a neural network:

$$\begin{aligned}\mathbf{e}_u^{(2)} = \omega\Bigg(\overbrace{\varphi\left(\mathbf{e}_i^{(1)}, \mathbf{e}_u^{(1)}\right)}^{(\square)}\Big\{\omega\Big(\Big\{\overbrace{\varphi(\mathbf{e}_{u'}, \mathbf{e}_i)}^{(\triangle)}\mathbf{e}_{u'},\\ \forall u' \in \mathcal{N}(i)\setminus\{u\}\Big\}\Big), \forall i \in \mathcal{N}(u)\Big\}\Bigg)\end{aligned} \tag{5.15}$$

that is, $\mathbf{e}_u^{(2)}$ depends on ($\square$) the importance each neighbor item node i has on the ego user node u after one hop, and ($\triangle$) the importance all users interacting with the same items as u have on their items. Note that ($\square$) may be further expanded:

$$\begin{aligned}\varphi\left(\mathbf{e}_i^{(1)}, \mathbf{e}_u^{(1)}\right) = \varphi\Big(\omega\Big(\Big\{\alpha_{u'\to i}^{(1)}\mathbf{e}_{u'}, \forall u' \in \mathcal{N}(i)\setminus\{u\}\Big\}\Big),\\ \omega\Big(\Big\{\alpha_{i'\to u}^{(1)}\mathbf{e}_{i'}, \forall i' \in \mathcal{N}(u)\setminus\{i\}\Big\}\Big)\Big)\end{aligned} \tag{5.16}$$

By making importance weights explicit, we get:

$$\varphi\Big(\mathbf{e}_i^{(1)}, \mathbf{e}_u^{(1)}\Big) = \varphi\Big(\omega\Big(\Big\{\varphi(\mathbf{e}_{u'}, \mathbf{e}_i)\mathbf{e}_{u'}, \forall u' \in \mathcal{N}(i) \setminus \{u\}\Big\}\Big), \omega\Big(\Big\{\varphi(\mathbf{e}_{i'}, \mathbf{e}_u)\mathbf{e}_{i'}, \forall i' \in \mathcal{N}(u) \setminus \{i\}\Big\}\Big)\Big) \tag{5.17}$$

When merging Equation (5.15) and Equation (5.17), it derives that:

$$\mathbf{e}_u^{(2)} = \omega\Big(\overbrace{\varphi\Big(\mathbf{e}_i^{(1)}, \mathbf{e}_u^{(1)}\Big)}^{\overbrace{\varphi(\mathbf{e}_{u'}, \mathbf{e}_i)}^{(\square)}\ \overbrace{\varphi(\mathbf{e}_{i'}, \mathbf{e}_u)}^{(\triangle)}} \Big\{\omega\Big(\Big\{\overbrace{\varphi(\mathbf{e}_{u'}, \mathbf{e}_i)}^{(\square)}\mathbf{e}_{u'}, \forall u' \in \mathcal{N}(i) \setminus \{u\}\Big\}\Big), \forall i \in \mathcal{N}(u)\Big\}\Big) \tag{5.18}$$

The node embedding for user u after two hops depends on ($\square$) the importance of all users interacting with the same items as u on those items, and ($\triangle$) the importance of all items interacted by u on user u. In other words, weighting the importance of each neighbor node on the ego node before the aggregation allows, after two propagation hops, to calculate to what extent each user profile is influenced by the profiles of the other users who rated the same items. Without loss of generality, a similar consideration could be made after a number of hops greater than two.

5.6.2 Enhancing neighborhood weighting through reviews

As known, graph-based models in machine learning are affected by oversmoothing [55, 295]. This phenomenon leads node embeddings, after multiple propagation hops, to become closer and closer in their representation in the latent space, eventually flattening their existing differences. As this behavior would profoundly weaken models' performance, exploration of the neighborhood generally tends to be constrained to very few hops (e.g., a maximum of two hops in attention-based weighting). However, in recommendation scenarios, limiting the exploration of the user-item bipartite graph may represent an inconsistency to the idea of collaborative filtering, where users are supposed to eventually be connected to share preferences and tastes for similar items.

Under this assumption, we believe the neighborhood weighting process could be further enhanced by exploiting other sources of information that are not usually taken into account. In the majority of popular online platforms for e-commerce (e.g., Amazon), *reviews* are fundamental tools to share *opinions* and *comments* about interacted items, as they convey the multi-faceted aspects that drove a user to interact with an item.

Leveraging such side information on the connections existing among users and items in the bipartite graph (i.e., graph edges) can improve the learning of the importance weights by reducing the oversmoothing effect because each user/item node embedding is conditioned on the informative and impression (i.e., opinion) conveyed by the review.

Let $\mathcal{W}_{ui} = \{w_1, w_2, \ldots, w_R\}$ be the set of R words that compose the review written by user u about item i. After an initial tokenization step, the sets of tokens for $\mathcal{W}_{ui}$ is defined as $\mathcal{T}_{ui} = \{t_1, t_2, \ldots, t_T\}$. Tokens are mapped to word embeddings, which are injected into an opinion-based model pretrained to predict the rating expressed by the user through specific terms in the review. While the output model carries the single information about the predicted review score, the activation of a hidden layer would unveil a richer source of textual features (i.e., an embedding) which drove the opinion-based model to predict that score. High-level features extracted from pretrained deep learning models can boost the recommendation performance of recommender systems leveraging items' side information (e.g., visual-based recommender systems [72, 107, 178]). We deem these textual features to deserve a pivotal role in this weighting process.

Let $\mathbf{r}_{ui} \in \mathbb{R}^f$ be the textual embedding extracted from the review of user u about item i through the pretrained opinion-based model. First, we project $\mathbf{r}_{ui} \in \mathbb{R}^f$ to the same latent space as $\mathbf{e}_u \in \mathbb{R}^d$ and $\mathbf{e}_i \in \mathbb{R}^d$ with a one-layer neural network:

$$\mathbf{p}_{ui} = \text{LeakyReLU}\left(\mathbf{W}\mathbf{r}_{ui} + \mathbf{b}\right) \tag{5.19}$$

where $\mathbf{p}_{ui} \in \mathbb{R}^d$ is the projected review embedding, while $\mathbf{W} \in \mathbb{R}^{f \times d}$ and $\mathbf{b} \in \mathbb{R}^d$ are the projection matrix and the bias, respectively. We seek to retain only those textual features of review $\mathbf{r}_{ui}$ which can be significant to later calculate the interdependence between this embedding and user/item ones.

Then, we propose to enhance the neighborhood weighting procedure at hop l by conditioning the importance weights also on the projected embedding of the review connecting user u and item i. For instance, the importance of the neighbor item node i on the ego user node u after l hops is calculated as:

$$\alpha_{i \to u}^{(l)} = \varphi\left(\mathbf{e}_i^{(l-1)}, \mathbf{e}_u^{(l-1)}, \mathbf{p}_{ui}\right) \tag{5.20}$$

Note that, since $\mathbf{p}_{ui}$ cannot increase the impact of the oversmoothing effect (because it is not dependent on the hop l), its usage in the importance weight formula becomes even more beneficial. Let us focus on the weighting function $\varphi(\cdot, \cdot, \cdot)$. Many approaches from the literature propose to leverage attention mechanisms, usually implemented

as a neural network trained in the downstream task to predict the importance of the neighbor node on the ego node. In our solution, we opt for a simplified and lightweight formulation that seeks to calculate the similarity between the neighbor and the ego nodes, conditioned on the opinion embedding of the review connecting them. Specifically:

$$\alpha_{i \to u}^{(l)} = \cos\left(\mathbf{e}_i^{(l-1)} \odot \mathbf{p}_{ui}, \mathbf{e}_u^{(l-1)} \odot \mathbf{p}_{ui}\right) \tag{5.21}$$

where $\odot$ is the element-wise multiplication, and $\cos(\cdot,\cdot)$ is the cosine similarity. Note that we suppress negative similarities to zero as such weights are usually non-negative [285]. Multiplying both node embeddings by the review opinion embedding provides the interplay between each node feature and the opinion features, thus producing a modified version of the node representation that conveys a richer source of information. It is worth pointing out that no trainable projection weight is learned in the presented formulation since the contribution of the review embedding is meaningful enough.

5.6.3 A double message-passing schema

The proposed neighborhood weighting procedure can help correct the representation error generated in the traditional message-passing schema. However, the idea is not to completely replace it, as several recent works from the literature have demonstrated its efficacy, especially in producing accurate recommendations [109]. The proposed approach involves a double message-passing schema, where two graph models are trained to refine their own user/item node representations. While the first one aggregates the contributions coming from the neighbor nodes into the ego nodes by weighting the neighborhood importance on the ego node *statically*, the second one aggregates the neighborhood's messages which are also weighted through the *opinion* embeddings from reviews.

We define the two graph convolutional networks as GCN_e (*error-affected*) and GCN_c (*correction*) and assign the node embeddings $\mathbf{e}_*$ to GCN_e, and the node embeddings $\mathbf{c}_*$ to GCN_c. As for the aggregation function, in both cases, we sum the weighted messages coming from the neighbor nodes. As such, the update of the user node embedding u

after l hops is calculated as:

$$\begin{aligned}
\text{GCN}_e \rightarrow \quad \mathbf{e}_u^{(l)} &= \sum_{i \in \mathcal{N}(u)} \alpha_{i \rightarrow u} \mathbf{e}_i^{(l-1)} = \sum_{i \in \mathcal{N}(u)} \frac{\mathbf{e}_i^{(l-1)}}{\sqrt{|\mathcal{N}(u)|}\sqrt{|\mathcal{N}(i)|}} \\
\text{GCN}_c \rightarrow \quad \mathbf{c}_u^{(l)} &= \sum_{i \in \mathcal{N}(u)} \alpha_{i \rightarrow u} \alpha_{i \rightarrow u}^{(l)} \mathbf{c}_i^{(l-1)} = \\
&= \sum_{i \in \mathcal{N}(u)} \frac{\cos\left(\mathbf{e}_i^{(l-1)} \odot \mathbf{p}_{ui}, \mathbf{e}_u^{(l-1)} \odot \mathbf{p}_{ui}\right)}{\sqrt{|\mathcal{N}(u)|}\sqrt{|\mathcal{N}(i)|}} \mathbf{c}_i^{(l-1)}
\end{aligned} \tag{5.22}$$

Note that $\alpha_{i \rightarrow u}$ is static and only depends on the topology of the bipartite graph, while $\alpha_{i \rightarrow u}^{(l)}$ varies along with the exploration hop and depends on the embeddings of ego/neighbor nodes, and the opinion review embedding. After L propagation hops, the final embedding representation is obtained as:

$$\begin{aligned}
\overline{\mathbf{e}}_u &= \sum_{l=0}^{L} \frac{1}{1+l} \mathbf{e}_u^{(l)}, \quad \overline{\mathbf{e}}_i = \sum_{l=0}^{L} \frac{1}{1+l} \mathbf{e}_i^{(l)} \\
\overline{\mathbf{c}}_u &= \sum_{l=0}^{L} \frac{1}{1+l} \mathbf{c}_u^{(l)}, \quad \overline{\mathbf{c}}_i = \sum_{l=0}^{L} \frac{1}{1+l} \mathbf{c}_i^{(l)}
\end{aligned} \tag{5.23}$$

where we apply the scaling factor $1/(1+l)$ to further alleviate the oversmoothing problem. A schematic overview of the node refining algorithm proposed for EGCF is displayed in Figure 5.6.

Given the learned *error-affected* and *correction* embeddings from above, EGCF predicts if a user u may interact with item i through the following formulation:

$$\hat{y}_{ui} = \underbrace{\overline{\mathbf{e}}_u^\top \overline{\mathbf{e}}_i}_{\text{error-affected}} + \underbrace{\overline{\mathbf{c}}_u^\top \overline{\mathbf{c}}_i}_{\text{correction}} \tag{5.24}$$

Thus, we apply the error correction to the user/item embedding representation only when predicting the user/item interaction. We optimize EGCF with the state-of-the-art Bayesian Personalized Ranking (BPR) [197].

5.6.4 Experimental Setup

Datasets. We use three popular [59, 162, 265] datasets from Amazon's Baby, Boys & Girls, and Men categories [182] which contain historical user-item interactions and reviews. We retain only interactions with non-empty reviews, then keep the 20k and

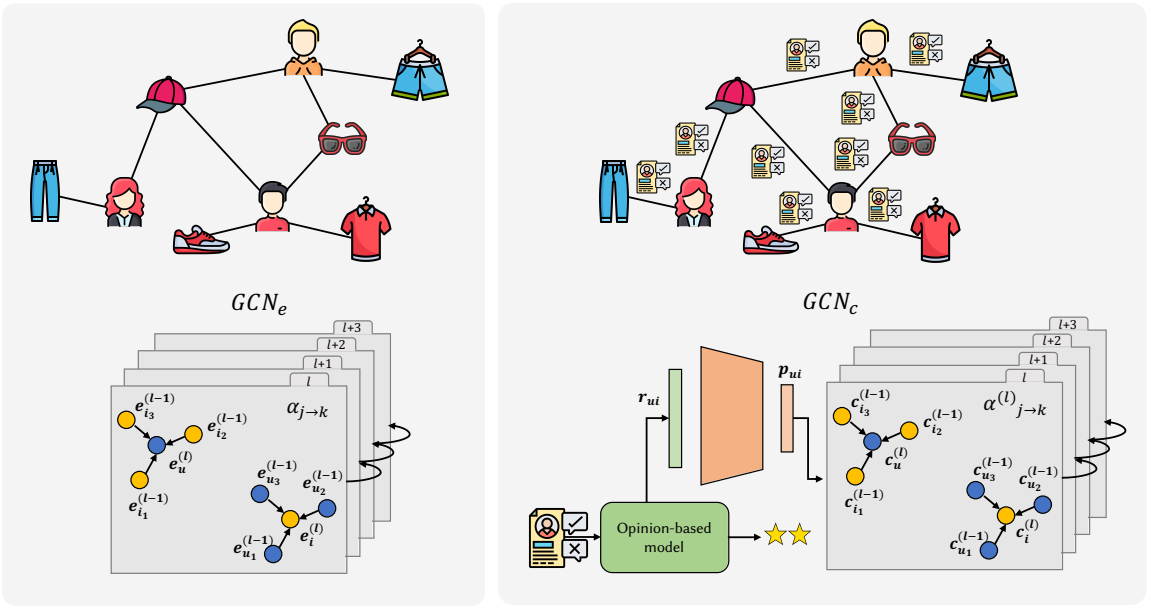


Fig. 5.6 Overview of the node refining algorithm proposed for EGCF. A statically-weighted GCN network affected by node representation error (a) is corrected through another GCN network (b), where an opinion-based embedding is extracted from each review as edge side information to weight the importance of the neighbor nodes on their ego nodes.

10k most popular items for Baby and Boys & Girls/Men, respectively. Finally, we apply the 5- and 15-core on items and users on Baby/Boys & Girls and Men, respectively. Final statistics are reported in Table 5.2.

Table 5.2 Statistics of the tested datasets.

Datasets	**#Users**	**#Items**	**#Interactions**	**Density**	**Average interactions per user**
Baby	4,669	5,435	29,214	0.00115	6.3
Boys & Girls	8,806	4,165	57,928	0.00158	6.6
Men	3,218	7,605	60,299	0.00246	18.7

Baselines. We compare our approach with eight state-of-the-art models spanning several families: (i) *traditional CF* (BPRMF [197] and MultiVAE [156]); (ii) *review-based CF* (ConvMF [135] and RMG [266]); (iii) *graph-based CF* (NGCF [261] and LightGCN [109]); (iv) *graph-based CF with attention* (GAT [252] and DGCF [262]).

Reproducibility. We adopt the temporal leave-one-out to split the datasets, where the last two recorded interactions are included in the validation and test. We tune hyper-parameters with [39] and follow the baselines papers, and fix the batch size to 256 and epochs to 400. As for EGCF, we extract review embeddings through a

popular pre-trained model[3]. Datasets and codes are publicly available[4]. All models are implemented in Elliot [15], our framework for reproducible recommender systems evaluation.

Evaluation protocol. We measure the model accuracy by adopting the recall (*Recall*@k), the normalized discounted cumulative gain ($nDCG@k$), and the mean average recall ($MAR@k$) [109, 262]. Additionally, considering the influence of novel and diverse recommendation lists [250, 251] on both user's and business's interests [3, 148], we also assess beyond-accuracy metrics such as the expected popularity complement ($EPC@k$) and the expected free discovery ($EFD@k$), along with indices measuring concentration and coverage, i.e., the 1's complement of the Gini ($Gini@k$), the Shannon entropy ($SE@k$), and the item coverage ($iCov@k$). Specifically, the $EPC@k$ and the $EFD@k$ refer to long-tail items and stand for the expected number of recommended unknown items which are also relevant, and the expected number of recommended known items which are also relevant, respectively. Furthermore, the $Gini@k$ and the $SE@k$ are used to assess items' distributional inequality, i.e., how unequally a recommender system shows different items to users, and the $iCov@k$ quantifies the number of items that the model recommends. For all metrics, higher values mean better recommendation performance.

5.6.5 Results and Discussion

Recommendation accuracy. Table 5.3 reports the results for accuracy measures on the top-10 recommendation lists. Surprisingly, the sole introduction of reviews does not seem to produce a consistent accuracy boost. For instance, the strongest review-based method (i.e., RMG) surpasses BPRMF only for the $nDCG$ and the MAR on Baby (i.e., 0.0911 vs. 0.0785 and 0.1059 vs. 0.0980, respectively). Contrarily, adopting a graph model can increase the accuracy to traditional CF. When comparing LightGCN with MultiVAE, which obtain the best performance in their respective recommendation families, we observe that the former improves, on Baby, the *Recall* of 7% and the MAR of 9%. However, the observed difference even reverts on Men for the $nDCG$ and the MAR. The application of attention mechanisms to weight the importance of neighbor nodes is rewarded in Baby and Boys & Girls, where GAT always outperforms NGCF, reaching remarkable results such as the *Recall* on Baby (i.e., 0.1595 vs. 0.1411) and the MAR on Boys & Girls (i.e., 0.1846 vs. 0.1783). Disentangling users' intents on interacted items (i.e., DGCF) produces even more accurate recommendations to NGCF

[3]https://huggingface.co/nlptown/bert-base-multilingual-uncased-sentiment.
[4]https://github.com/sisinflab/Edge-Graph-Collaborative-Filtering.

Table 5.3 Calculated accuracy metrics, i.e., *Recall*, *nDCG*, and *MAR*, for top-10 lists. Best value is in **bold**, while second-to-best is underlined.

Models	Baby			Boys & Girls			Men		
	Recall	*nDCG*	*MAR*	*Recall*	*nDCG*	*MAR*	*Recall*	*nDCG*	*MAR*
MostPop	0.0940	0.0520	0.0627	0.1195	0.0647	0.0776	0.0702	0.0590	0.0672
BPRMF	0.1377	0.0785	0.0980	0.1821	0.1446	0.1666	0.1662	0.1314	0.1527
MultiVAE	0.1768	0.1262	0.1455	0.2224	0.1695	0.1990	0.2091	0.1656	0.1898
ConvMF	0.1230	0.0647	0.0800	0.1146	0.0831	0.0972	0.0838	0.0524	0.0584
RMG	0.1272	0.0911	0.1059	0.1512	0.1065	0.1325	0.1067	0.0727	0.0867
NGCF	0.1411	0.0916	0.1092	0.2006	0.1523	0.1783	0.1969	0.1461	0.1722
LightGCN	0.1892	0.1362	0.1590	0.2305	0.1743	0.2054	0.2124	0.1605	0.1882
GAT	0.1595	0.1051	0.1233	0.2069	0.1573	0.1846	0.1695	0.1254	0.1476
DGCF	0.1874	0.1352	0.1558	0.2249	0.1716	0.2023	0.2070	0.1554	0.1823
EGCF	**0.1944***	**0.1402***	**0.1623***	**0.2325**	**0.1792***	**0.2089***	**0.2195***	**0.1703***	**0.1988***

***statistically significant differences (p-value* ≤ 0.05*).*

on all datasets. Nevertheless, LightGCN always performs better than DGCF apart from very few cases (i.e., *nDCG* and *MAR* on Men), even though DGCF's calculated accuracy values do not substantially differ from LightGCN's ones (e.g., see the *MAR* on Baby). Noticeably, the proposed model (i.e., EGCF) outperforms the other baselines under all settings and datasets, with near 100% statistical hypothesis tests (i.e., paired t-test) showing that the results significantly differ. This finding further motivates the goodness of the solution. While we observe a substantial accuracy improvement in traditional and review-based approaches (e.g., +12% to MultiVAE for the *MAR* on Boys & Girls and +53% to RMG for the *Recall* on Baby), introducing an additional GCN-like network guided by users' reviews is even more beneficial to correct the representation error observable in unweighted graph approaches. Particularly, results show that such correction may lead to small accuracy improvements in some cases (e.g., see the *Recall* on Boys & Girls when correcting LightGCN) but also larger ones in other cases (e.g., see the *nDCG* on Men when correcting LightGCN). Such outcomes suggest that while keeping the error-affected contribution in the final prediction formula is useful to preserve the superior performance of graph-based models to traditional and review-based approaches, the introduced correction term is useful to gain even more accurate preference predictions than unweighted graph architectures.

Recommendation novelty and diversity. We also assess how novel and diverse recommendation lists are. The two novelty metrics in Table 5.4 (i.e., the *EPC*@k and the *EFD*@k) are discussed with concentration and coverage indices in Table 5.5 (i.e., the *Gini*@k, the *SE*@k, and the *iCov*@k) as in an ideal recommender system, a

Table 5.4 Calculated novelty metrics, i.e., EPC and EFD, for top-10 lists. Best value is in **bold**, while second-to-best is underlined.

Models	Baby		Boys & Girls		Men	
	EPC	EFD	EPC	EFD	EPC	EFD
MostPop	0.0108	0.0728	0.0135	0.0913	0.0112	0.0904
BPRMF	0.0164	0.1153	0.0306	0.2282	0.0259	0.2167
MultiVAE	0.0268	0.2088	0.0360	0.2874	0.0333	0.2912
ConvMF	0.0135	0.0930	0.0174	0.1219	0.0102	0.0857
RMG	0.0193	0.1488	0.0226	0.1787	0.0144	0.1226
NGCF	0.0194	0.1463	0.0323	0.2510	0.0292	0.2531
LightGCN	0.0289	0.2271	0.0371	0.3012	0.0323	0.2856
GAT	0.0223	0.1708	0.0334	0.2616	0.0248	0.2106
DGCF	0.0287	0.2228	0.0365	0.2945	0.0311	0.2734
EGCF	**0.0298***	**0.2359***	**0.0382***	**0.3120***	**0.0343***	**0.3066***

**statistically significant differences (p-value* ≤ 0.05*)*

loosely concentrated and large set of recommended items should equally span different ranges of popularity. As previously observed, EGCF is again the best or second-to-best technique. While NGCF is not as capable as LightGCN of proposing long-tail items on Boys & Girls (e.g., 0.2510 vs. 0.3012 for the EFD), the former surpasses the latter for the concentration indices on the same dataset (e.g., 10.5595 vs. 10.1586 for the SE). Since NGCF adopts an ego-neighbor interaction component, the concentration of explored and recommended near items gets loose. Moreover, neighborhood weighting leads to recommend items from the long tail (e.g., comparing GAT with NGCF, we observe a +17% for the EFD on Baby). However, such a finding is not consistent with the trend recognized for the concentration and coverage indices (e.g., when comparing LightGCN with DGCF, we notice 0.1304 vs. 0.2051 for the *Gini* on Men), as the neighborhood weighting procedure comes at the expense of a limited hop exploration, not allowing such models to explore wider catalog portions. Conversely, injecting user-generated reviews brings new informative content (e.g., RMG recommends a broader and less concentrated range of items from the catalog than DGCF on the Baby dataset). Finally, weighting the neighborhood importance and exploring long-distant user-item interactions through reviews-enriched content (i.e., EGCF) allows to retrieve larger portions of heterogeneous items (e.g., EGCF outperforms LightGCN for the *Gini* by +63% on Baby and DGCF for the SE by +7% on Boys & Girls), without retaining less popular items from the long-tail (observing the same models, +3% for the EPC on Baby and +6% for the EFD on Boys & Girls). Such outcomes demonstrate that the content enrichment brought by the extracted review features (injected into

Table 5.5 Calculated concentration and coverage indices, i.e., *Gini*, *SE*, and *iCov*, for top-10 lists. Best value is in **bold**, while second-to-best is underlined.

Models	Baby			Boys & Girls			Men		
	Gini	*SE*	*iCov*	*Gini*	*SE*	*iCov*	*Gini*	*SE*	*iCov*
MostPop	0.0018	3.5313	18	0.0023	3.5724	18	0.0015	3.9332	32
BPRMF	0.0019	3.7819	40	0.0031	4.0921	203	0.0037	5.2991	192
MultiVAE	0.2139	**9.9160**	4,143	0.2671	10.2463	3,824	0.1085	9.8988	3,014
ConvMF	0.0018	3.5933	18	0.0030	3.9745	220	0.0029	4.6783	265
RMG	0.1059	9.4892	2,130	0.1567	9.7193	2,538	0.1146	10.0344	2,549
NGCF	0.0948	8.8700	2,641	0.3031	**10.5595**	3,668	0.1749	10.7116	3,651
LightGCN	0.1405	9.3105	3,417	0.2398	10.1586	3,647	0.2051	10.8815	4,384
GAT	0.1370	9.2024	3,102	0.2496	10.2821	3,449	0.1235	9.7802	3,530
DGCF	0.0673	8.3193	2,325	0.1800	9.7617	3,208	0.1304	10.2011	3,378
EGCF	**0.2294**	9.8535	**4,490**	**0.3037**	10.4545	**4,030**	**0.2208**	**10.8876**	**4,920**

the representation error correction) allows to explore user-item interactions at multiple hops, leading to more heterogeneous recommendation lists which also include items from the long-tail.

Effect of hop exploration number. Figure 5.7 displays, for EGCF, the *Recall*@k and EFD@k performance variation on top-10 recommendation lists when exploring a number of hops in the range 1-4, where even numbers stand for same node type connections (e.g., user-user), while odd numbers refer to opposite node type connections (i.e., user-item). As evident from the histograms of Baby and Boys & Girls, the *Recall*@k consistently increases from 1 to 4 hops (this is why we adopt four hop explorations for EGCF on those datasets). The same trend is not observable for Men, where two explored hops seem to provide the highest accuracy boost, motivating the adoption of 2 hop explorations for EGCF on the same dataset. Such behavior could be due to the average number of users' interacted items in Men (approximately 19, see Table 4.4). The node refining probably does not require a broad exploration of its neighborhood. As for the EFD@k, the Baby and the Men datasets seem to agree on two exploration hops to produce the most diverse item lists of recommendations because they leverage (as previously recalled) user-user and item-item interconnections (and similarities). The trend is also aligned with the Boys & Girls dataset, where user-user and item-item links are exploited even at a higher depth (i.e., four exploration hops). The emerged insights shed light on two main contributions: (i) with the modified neighborhood weighting process, which makes use of reviews to enhance the informative content carried by user-item interactions, EGCF is less limited in the

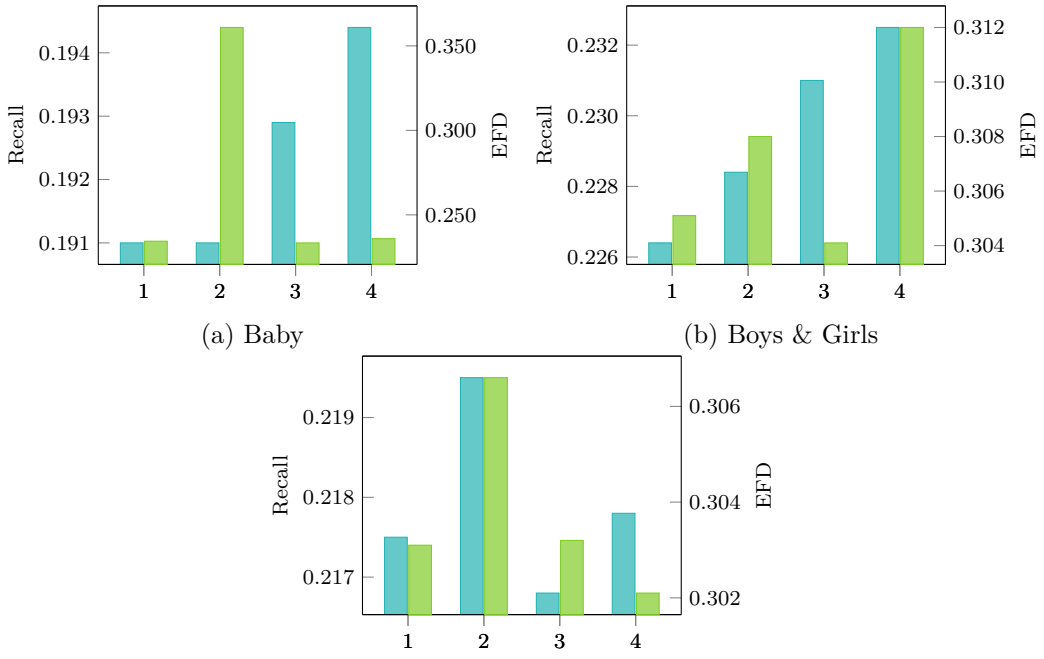


Fig. 5.7 Recommendation performance of EGCF, i.e., *Recall@k* (histogram bars in teal blue) and *EFD@k* (histogram bars in lime green), on top-10 recommendation lists, when varying the number of explored hops from 1 to 4.

hop exploration, thus providing more accurate recommendations, and (ii) user-user and item-item connections are the keystones on which building more diverse item recommendation lists.

5.7 Summary

This chapter investigates the performance of graph collaborative filtering models on beyond accuracy metrics showing that their superior accuracy capabilities is reached at the expense of user fairness, item exposure, and their combination. The nodes representation and neighborhood exploration emerge as the main dimensions to categorize graph CF solutions, and the study complements with an analysis of their influence on beyond accuracy measures and overall accuracy separately and simultaneously. The analysis not only confirms the experimental results and raises concerns about the effective application of recent approaches in graph CF (e.g., beyond message-passing). Given the presented results in the first part of this chapter, we studied the accuracy/novelty/diversity trade-offin graph collaborative filtering for different neighborhood exploration strategies (i.e., explicit and implicit message-passing) and

depths (i.e., number of explored hops). Results for six state-of-the-art graph models on three e-commerce datasets reveal that the accuracy/diversity trade-offis reachable only when explicitly propagating messages. We show that user-user and item-item explorations may improve accuracy/diversity/novelty trade-offthanks to a message-passing reformulation. We plan to expand the evaluation to recent graph models which optimize diversity and better investigate the same-type node setting. Finally, we propose Edge Graph Collaborative Filtering (EGCF), which incorporates users' opinions extracted from reviews into the edges of a GCN to weight the neighborhood importance on the ego node. Extensive experimental evaluation shows that EGCF outperforms traditional, review- and graph-based models. The work complements with an analysis of beyond-accuracy performance and an extensive study on the number of layers. We plan to: *(i)* improve the representation of nodes via the use of different similarities, and *(ii)* integrate the diverse GCNs into a single hyper-GCN.

Chapter 6

Explicable Recommendations (post-hoc): critical issues and new proposals

Topic related to explainable recommendations is certainly not new, but it has recently come back into vogue in part because of a growing interest in the machine learning community with respect to this issue. In fact, the research field of eXplainable Artificial Intelligence (XAI) has been increasingly successful influencing even neighboring fields, not only for computer science but also social sciences, philosophy and the legal side.

In terms of influence concerning recommender systems, recent studies have focused on making the recommendation process explainable through Rule Mining or Topic Modeling mechanisms. In contrast, others have focused on exploiting a knowledge base's content to bridge this gap. Another branch of interest is the one that has tried to obtain explainable recommendations by integrating attention mechanisms into the models to try to justify the explanation produced by the model concerning some characteristics of the client or some of past "actions" (click, review, purchased) or even considering the attributes of the suggested items. In parallel, some researchers began to wonder whether it might not be appropriate to separate the explanation task from the recommendation task, thus drawing inspiration from Post-hoc (or model agnostic) approaches typical of the XAI field to adapt them to the recommendation task.

Our investigation showed that this approach to generating explanations for proposed suggestions is particularly crucial. Developing post-hoc explanations certainly does not affect the recommendation model; conversely, it requires that the user modeling underlying the recommendation process be very robust. In addition, the explanation task must be accompanied by some constraints that may or may not be revealed by the

system. This aspect, which goes to privacy, is often overlooked but plays a strategic role in understanding what types of information are disclosable to the actors involved in a recommendation process. As such, our contribution will provide a formal approach that can support explanations for a Multi-Stakeholder recommendation context.

6.1 Introduction

Generating explanations with a Post-hoc approach makes the explanation process independent of the actual recommendation process. It also makes the explanation process applicable to any recommendation model that is considered as a magic box that generates suggestions. However, applying these types of approaches introduces additional complexity within the process. Indeed, one has to train an explanation model separately from the recommendation model and test it separately. Often, the applicability of well-known techniques in the XAI literature are not easily applied to recommendation models, especially large-scale ones. Typically, the problems that Post-hoc approaches suffer from (such as computational complexity) are amplified when applying these approaches to recommendation systems.

Among the Post-hoc explanation approaches that have become well established in the XAI community is definitely the SHAP approach [165] based on Shapley Values. Although this approach is based on solid formal foundations, it has the serious limitation of becoming computationally unviable because of the large number of elements in the dataset or because of the high dimensionality of the recommendation model it attempts to explain. Some attempts have been made to make this type of solution applicable even in unfavorable cases, for example, attempts have been made to alleviate the problem by sampling the elements involved or by adopting solutions that approximate the Shapley value problem on which this explanation mechanism is based. Recently, an approach called FastSHAP [129] has been proposed that aims to make the SHAP explanation approach computationally efficient without losing in terms of performance (compared to the task of reconstructing the exact Shapley values).

In the context of post-hoc approaches based on surrogate models, LIME [208] indeed represents one of the models that has found the most success in recent years. Although attributable to a particular case of SHAP explanation, LIME is an algorithm that can faithfully explain the predictions of any classifier or regressor by approximating it locally with an interpretable model. As stated before, LIME belongs to the category of post-hoc algorithms and it sees the prediction system as a black box by ignoring its underlying operations and algorithms. Since we can consider the recommendation

task as a particular Machine Learning task, the LIME approach can also be applied to recommendation. LIME-RS [186] is an adaptation of the general algorithm to the recommendation task and can be considered in all respects as a black-box explainer. This means that it generates an explanation by drawing a huge number of (random) calls to the system, collecting the answers, building a model of behavior of the system, and then constructing the explanation for the particular recommended item. Given the explanation that a system can provide to a user we identify at least two characteristics that the explanation part should enforce [96, 180, 243]:

- *Adherence* to reality: the explanation should mention only features that really pertain to the recommended item. For instance, if the system recommends the movie "Titanic", it should not explain this recommendation by saying "because it is a War Movie" since it is by no means an adherent description of that movie;

- *Constancy* in the behavior: when the explanation is generated based on some sample, and such a sample is drawn with a probability distribution, the entire process should not exhibit a random behavior to the user. For instance, if the explanation for recommending the movie "The Matrix" to the same user is first "because it is a Dystopian Science Fiction", and then "because it is an Acrobatic Duels Movie", for the same user, this behavior would be perceived as nondeterministic, and thus reducing its trustworthiness.

While the fact of adopting a black-box approach lets LIME-RS to be applicable for every recommender system, the way of building a model by drawing a huge random sample of system behaviors makes it lose both adherence and constancy, as our experiments show later on this chapter. This suggests that the direct application of LIME-RS to recommender systems is not advisable, and that further research is needed to assess the usefulness of LIME-RS in explaining recommendations.

6.2 Related Work

In recent years, the theme of Explanation in Artificial Intelligence has come to the foreground, capturing the attention not only of the Machine Learning and related communities – that deal more specifically with the algorithmic part – but also of fields closer to Social Sciences, such as Sociology or Cognitivism, which look with great interest to this area of research [180]. The growing interest in this area is also dictated by new regulations of both Europe [256] and US [54] with respect to sensitive issues in the field of personal data processing, and legal responsibility. This trend has also

touched the research field of recommender systems [26, 188, 193, 289]. However, topics such as explanation are by no means new to this field. In fact, we can date back to 2014 the introduction of the term "explainable recommendation" [290], although the need to provide an explanation that accompanies the recommendation is a need that emerged as early as 1999 by Schafer et al. [222], when people began trying to explain a recommendation with other similar items familiar to the user who received that recommendation.

Catalyzation of interest around the topic of explanation of recommendations coincides also with the awareness achieved in considering metrics beyond accuracy as fundamental in evaluating a recommendation system [177, 250]. Indeed, all of the well-known metrics of novelty, diversity, and serendipity are intended to improve the user experience, and in this respect, a key role is played by explanation [96, 242]. "Why are you recommending that?"—this is the question that usually accompanies the user when a suggestion is provided. Tintarev et al. [243] detailed in a scrupulous way the aspects involved in the process of explanation when we talk about recommendation. They identified 7 aspects: user's trust, satisfaction, persuasiveness, efficiency, effectiveness, scrutability, and transparency.

This is the starting point to define Explainable Recommendation as a task that aims to provide suggestions to the users and make them aware of the recommendation process, explaining also why that specific object has been suggested. Gedikli et al. [96] evaluated different types of explanations and drew a set of guidelines to decide what the best explanation that should equip a recommendation system is. This is due to the fact that popular recommendation systems are based on Matrix Factorization (MF) [141]; for this type of model, trying to provide an explanation opens the way to new challenges [59, 63, 180, 246].

There are two different approaches to address this type of issue.

- On the one hand, the model-intrinsic explanation strategy aims to create a user-friendly recommendation model or encapsulates an explaining mechanism. However, as Lipton [160] points out, this strategy will weigh in on the trade-off between the transparency and accuracy of the model. Indeed, if the goal becomes to justify recommendations, the purpose of the system is no longer to provide only personalized recommendations, resulting in a distortion of the recommendation process.

- On the other hand, we have a model-agnostic [263] approach, also known as post-hoc [191], which does not require to intervene on the internal mechanisms

of the recommendation model and therefore does not affect its performance in terms of accuracy.

Most recommendation algorithms take an MF-approach, and thus the entire recommendation process is based on the interaction of latent factors that bring out the level of liking for an item with respect to a user. Many post-hoc explanation methods have been proposed for precisely these types of recommendation models. It seems evident that the most difficult challenge for this type of approach lies in making these latent factors explicit and understandable for the user [26]. Peake et al. [191] generate an explanation by exploiting the association rules between features; Tao et al. [240] in their work, find benefit from regression trees to drive learning, and then explain the latent space; instead, Gao et al. [95] try a deep model based on attention mechanisms to make relevant features emerge. Along the same lines are Pan et al. [188], who present a feature mapping approach that maps the uninterpretable general features onto the interpretable aspect features. Among other approaches to consider, [290] proposes an explicit factor model that builds a mapping between the interpretable features and the latent space. On the same line we also find the work by Fusco et al. [91]. In their work, they provide an approach to identify, in a neural model, which features contribute most to the recommendation. However, these post-hoc explanation approaches turn out to be built for very specific models. Purely model-agnostic approaches include the recent work of Tsang et al. [244], who present GLIDER, an approach to estimate interactions between features rather than on the significance of features as in the original LIME [208] algorithm. This type of solution is constructed regardless of the recommendation model.

Our investigation focuses on the operation of LIME, a model-agnostic method for a surrogate-based local explanation. When a user-item pair is provided, this model returns as an outcome of the explanation a set of feature weights, for any recommender system. However, the recommendation task is very specific, so there is a version called LIME-RS [186] that applies the explanation model technique to the recommendation domain. In this way, any recommender is seen as a black box, so LIME-RS plays the role of a model-agnostic explainer whose result is a set of interpretable features and their relative importance.

The goal of LIME-RS is to exploit the predictive power of the recommendation (black box) model to generate an explanation about the suggestion of a particular item for a user. In this respect, it exploits a neighborhood drawn according to a generic distribution compared to the candidate item for the explanation. It seems obvious that the choice of the neighborhood plays a crucial role within the process of

explanation generation by LIME-RS. We can compare this sample extraction action to a perturbation of the user-item pair we are using to generate the explanation. In the case of LIME-RS this perturbation must generate consistent samples with respect to the source dataset. We see that this choice represents a critical issue for all the post-hoc models which base their expressiveness on the locality of the instance to explain.

This trend is confirmed in several papers addressing this issue of surrogate-based explanation systems such as LIME and SHAP [233]. In two recent works, Alvarez-Melis et al. [11] have shown how the explanations generated with LIME are not very robust: their contribution aims to bring out how small variations or perturbations in the input data cause significant variations in the explanation of that specific input [10]. In their paper, a new strategy is introduced to strengthen these methods by exploiting local Lipschitz continuity. By deeply investigating this drawback, they introduced self-explaining models in stages, progressively generalizing linear classifiers to complex yet architecturally explicit models.

Saito et al. [216] also explored this issue by turning their gaze to different types of sampling to make the result of an explanation generated through LIME more robust. In particular, in their work, they introduce the possibility of generating realistic samples produced with a Generative Adversarial Network. Finally, Slack et al. [227] adopt a similar solution in order to control the perturbation generating neighborhood data points by attempting to mitigate the generation of unreliable explanations while maintaining a stable black-box model of prediction.

6.3 Background

From a formal point of view, we can define a LIME-generated explanation for a generic instance $x \in \mathcal{X}$ produced by a model f as:

$$\xi(x) = \underset{e \in E}{\operatorname{argmin}} \mathcal{L}(f, e, \pi_x) + \Omega(e) \tag{6.1}$$

where $\mathcal{L}$ represents the fidelity of the surrogate model to the original f, and e represents a particular instance of the class E of all possible explainable models. Among all the possible models, the one most frequently chosen is based on a linear prediction. In this case, an explanation refers to the weights of the most important interpretable features, which, when combined, minimize the divergence from the black-box model. The function π_x measures the distance between the instance to be explained $x \in \mathcal{X}$,

and the samples $x' \in \mathcal{X}$ extracted from the training set to train the model e. Finally, $\Omega(e)$ represents the complexity of the explanation model.

Two pieces of evidence make the application of LIME possible: (i) the existence of a feature space $\mathcal{Z}$ on which to train the surrogate model of f, (ii) and the presence of a surjective function that maps the space mentioned above ($\mathcal{Z}$) to the original space of instances ($\mathcal{X}$). Going into more detail, we consider the fidelity function $\mathcal{L}$ as the mean square deviation between the prediction for a generic instance $x' \in \mathcal{X}$ of the black-box model and that generated for the counterpart $z' \in \mathcal{Z}$ by the surrogate model. Starting from these considerations we can express $\mathcal{L}$ with the following formula:

$$\mathcal{L}(f, e, \pi_x) = \sum_{x' \in \mathcal{X}, z' \in \mathcal{Z}} \pi_x\left(x'\right) \cdot \left(f\left(x'\right) - e\left(z'\right)\right)^2 \tag{6.2}$$

In the formula above π_x plays a fundamental role as it expresses the distance between the instance to be explained and the sampled instance used to build the surrogate model. From a generic perspective, we can express this function as a kernel function like $\pi_x = exp(-D(x, x')^2/\sigma^2)$, where D is any measure of distance.

The full impact of this distance is captured when the fidelity function also considers the transformation of the surrogate sample in the original space. As mentioned earlier, we consider a surjective function p that maps the original space into the feature space $p : \mathcal{X} \to \mathcal{Z}$. We can also consider the function that allows us to move in the opposite direction $p^{-1} : \mathcal{X} \to \mathcal{Z}$. At this point, Equation (6.2) becomes:

$$\mathcal{L}(f, e, \pi_x, p) = \sum_{z' \in \mathcal{Z}} \pi_x\left(p^{-1}\left(z'\right)\right) \cdot \left(f\left(p^{-1}\left(z'\right)\right) - e\left(z'\right)\right)^2 \tag{6.3}$$

From this last equation, we can grasp the criticality of the surjective mapping function. Indeed, the neighborhood in $\mathcal{Z}$-space cannot be guaranteed with the transformation in $\mathcal{X}$-space. Thus, some samples selected to train the surrogate model could not satisfy the neighborhood criterion for which they were chosen.

We must therefore stress on the centrality of the sampling function: how do we extract the neighborhood of our instance to be explained? If we look at the application of LIME to the recommendation domain, we can compare this sampling action to a local perturbation around our instance x; however, this perturbation aims to generate n samples x', which might contain inconsistencies: as an example, suppose we want to explain *James*'s feeling about the movie *The Matrix*. The original triple of the instance to be explained associates to the user-item pair the genre of the movie (representing the explainable space) and in this case it is of the type $\langle James, TheMatrix, Sci\text{-}Fi \rangle$. A perturbation around this instance could generate

inconsistencies of the type $\langle James, TheMatrix, Western \rangle$. For this reason, in LIME-RS the perturbation considers only real and not synthetic data. This choice is dictated by the avoidance of the out-of-sample (OOS) process phenomenon. Closely related to this problem predicted by OOS is that the interpretation examples selected in LIME-RS represent the ability to capture the locality through disturbance mechanisms effectively. One of the disadvantages of LIME-like methods is that they sometimes fail to estimate an appropriate local replacement model but instead generate a model that focuses on explaining the examples and is also affected by more general trends in the data.

This issue is central to our work, and it involves two aspects: (i) the first one concerns the sampling function of the samples precisely. In the LIME-RS implementation, this function is driven by the popularity distribution of the items within the dataset. (ii) The second critical issue concerns the model's ability to wittily discriminate the user's taste from the neighborhood extracted to build the surrogate model. A model that squashes too much on bias or is inaccurate cannot bring out the peculiarities of user taste that are critical in building the explainable model which are, in turn, useful in generating the explanation for the instance of interest.

These observations dictate the two questions that motivated our work: can we consider the surrogate-based model on which LIME-RS is built to generate always the same explanations, or does the extraction of a different neighborhood severely impact the system's constancy? In addition, are LIME-RS explanations adherent to item content, despite the fact that the sampling function is uncritical and based only on popularity?

6.4 Experiments

This section is devoted to illustrating how the experimental campaign was conducted. The datasets used for this phase of experimentation are *Movielens 1M* [105], *Movielens Small* [105], and *Yahoo! Movies*[1]. Their characteristics are shown in Table 6.1.

Table 6.1 Characteristics of the datasets involved in the experiments.

	#Users	**#Items**	**#Interactions**	**Sparsity**
Movielens 1M	6040	3675	797758	0,9640
Movielens Small	610	8990	80419	0,9853
Yahoo! Movies	7636	8429	160604	0,9975

[1]R4 - Yahoo! Movies User Ratings and Descriptive Content Information, v.1.0 http://webscope.sandbox. yahoo.com/.

As for the choice of the models to be used in this work is concerned, we selected two well-known recommendation models that are able to exploit the information content of the items to produce a recommendation: Attribute Item kNN (Att-Item-kNN) and Vector Space Model (VSM). The two chosen models represent the simplest solution to address the recommendation problem by exploiting the content associated with the items in the catalog.

Att-Item-kNN exploits the characteristics of neighborhood-based models but expresses the representation of the items in terms of their content and, based on this representation, it computes a similarity between users. Starting from this similarity and exploiting the collaborative contribution in terms of interactions between users and items, Att-Item-kNN tries to estimate the level of liking of the items in the catalog. VSM represents both users and items in a new space to link users and items to the considered information content. Once obtained this new representation, with an appropriate function of similarity, VSM estimates which are the most appealing items for a specific user. The implementation of both models are available in the ELLIOT [15] evaluation framework. This benchmarking framework was used to select the best configuration for the two recommendation models by exploiting the corresponding configuration file[2].

Our experiments start by selecting the best configurations based on nDCG [25, 143] for the two models on the considered datasets. Then, we generate the top-10 list of recommendations for each user, and we take into account the first item i_1 on these lists for each user u. Finally, each recommendation pair (u, i_1) is explained with LIME-RS. The explanation consists of a weighted vector $(g, w)_i$ where g is the genre of the movies in the dataset – *i.e.*, the features – and w is the weight associated to g by LIME-RS within the explanation. Then, this vector is sorted by descending weights. In this way, the genres of the movies which play a key role within the recommendation, as explained by LIME-RS, are highlighted at the first positions of the vector. These operations are then repeated $n = 10$ times and changing the seed each time, as 10 is likely to be a good choice to detect a general pattern in the behavior of LIME-RS. At this point, for each pair (u, i_1), we have a group of 10 explanations ordered by descending values of w, which we exploit to answer our questions.

[2]https://tny.sh/basic_limers

6.4.1 Results

Empirically, since in a real scenario of recommendation a too verbose explanation is not useful, we consider only the first five features in the sorted vector representing the explanation of each recommendation. In order to verify the constancy of the behavior of LIME-RS, given a (u, i_1) pair, we exploit the n previously generated explanations for this pair. Then for $k = 1, 2, \ldots, 5$, we define G_k as the multiset of genres that appear in k-th position – for instance, if "Sci-Fi" occurs in the first position of 7 explanations, then "Sci-Fi" occurs 7 times in the multiset G_1, and similarly for other genres and multisets. Then, we compute the frequency of genres in each position as follows: given a position k, a genre g, and the number n of generated explanations for a given pair (u, i_1), the frequency f_{g_k} of g in k-th position is computed as:

$$f_{g_k} = \frac{||\{g \mid g \in G_k\}||}{n} \tag{6.4}$$

where $||\cdot||$ denotes the cardinality of a multiset. Then, all this information is collected for each user in five lists — one for each of the k positions — of pairs $\langle g, f_{g_k} \rangle$ sorted by frequency. One can observe that the computed frequency is an estimation of the probability that a given genre is put in that position within the explanation generated by LIME-RS sorted by values. Hence, the pair $\langle g, \max(f_{g_k}) \rangle$ describes the genre with the highest frequency in the k-th position of the explanation for a pair (u, i_1). Finally, it makes sense to compute the mean μ_k of the highest probability values in each position k of the explanations for each pair (u, i_1). Formally, by setting a position k, the mean μ_k is computed as:

$$\mu_k = \frac{\sum_{j=1}^{|U|} \max(f_{g_k})_j}{|U|} \tag{6.5}$$

where U is the set of users for whom it was possible to generate a recommendation for. Observing the value of μ_k, we can state to what extent LIME-RS is constant in providing the explanations until the k-th feature: the higher the value of μ_k, the higher the constancy of LIME-RS concerning the k-th feature.

By looking at Table 6.2, we can see that for Att-Item-kNN the LIME-RS explanation model is reliable as long as it considers at most three features in the weighted vector presented as an explanation of the recommendation. Extending the explanation to four features, we have a constancy that falls below 65%, while arriving at an explanation with five features is more likely to run into explanations that exhibit an unacceptably random behavior. On the other hand, we can see that for VSM the values are much more stable. In this case, we have a constancy that, regardless of the length of the

Table 6.2 Constancy of LIME-RS. A value equal to 0 means that the genre(s) provided by LIME-RS in the first k position(s) is always different (worst case: completely inconstant behavior); A value equal to 1 means that the genre(s) provided by LIME-RS in the first k position(s) is always the same (total constancy).

	μ_1	μ_2	μ_3	μ_4	μ_5
Att-Item-kNN					
Movielens 1M	0,9130	0,7822	0,6927	0,6288	0,5727
Movielens Small	0,8830	0,7426	0,6639	0,60459	0,5616
Yahoo! Movies	**0,9230**	**0,8016**	**0,7232**	**0,6528**	**0,5830**
VSM					
Movielens 1M	0,8929	0,7953	0,7729	0,7726	0,7801
Movielens Small	0,9464	0,8636	0,8343	0,8138	0,8049
Yahoo! Movies	**0,9732**	**0,9209**	**0,8887**	**0,8884**	**0,9056**

weighted vector of the explanation, stabilizes on average around 80%. An aspect emerges that will be discussed in detail later: LIME-RS is conditioned by the ability of the black-box model to discriminate the user's tastes locally.

With the aim of providing an answer about the adherence to reality of LIME-RS, we make a comparison between the genres claimed to explain a recommended item and its actual genres. Indeed, the explanations about an item should fit the list of genres the item is characterized by. This means that, in an ideal case, all highly weighted features within the explanation should match the genres of the item. From the results in Table 6.2, we notice that using Att-Item-kNN the constancy of LIME-RS reaches a low value after the third feature. Hence, it is a futile effort to go deeper in the study of the explanation. To this aim, we intersected each explanation limited to the set E_k of its first k genres with the set of genres F_{i_1} characterizing the first recommended item, for $k = 1, 2, 3$. Upon completion of this operation for all the n explanations generated for each (u, i_1) pair, we computed the number of times we obtained an empty intersection of these sets, normalized by the total number of explanations $n \times |U|$, in order to understand to what extent an explanation is (not) adherent to the item. Formally, for a given value of k, the value $adherence_k$ is computed as:

$$adherence_k = \frac{\sum_{j=1}^{n \times |U|} \left[(E_k \cap F_{i_1})_j = \emptyset \right]}{n \times |U|} \tag{6.6}$$

where U is the set of users of the dataset for whom it was possible to generate a recommendation, n is the number of generated explanations for each pair (u, i_1), and by $\Sigma[\cdots]$ we mean that we sum 1 if the condition inside $[\cdots]$ is true, and 0 otherwise. One can note that $adherence_k \in [0, 1]$, where a value of 1 indicates the worst case in

which for none of the n explanations under consideration at least one genre of the item is in the first k features of the explanation. In contrast, the lower the value of $adherence_k$, the higher the adherence of LIME-RS.

Table 6.3 Adherence of LIME-RS. For value equals to 1 no genre provided by LIME-RS in the first k real genres of the movie (worst case); For value equals to 0 at least one genre provided by LIME-RS in the first k genres is always among the real genres of the movie.

	$adherence_1$	$adherence_2$	$adherence_3$
Att-Item-kNN			
Movielens 1M	0,2774	0,1105	0,0488
Movielens Small	**0,2364**	**0,0651**	**0,0180**
Yahoo! Movies	0,3597	0,1202	0,0476
VSM			
Movielens 1M	0,5357	0,2539	0,1088
Movielens Small	0,4384	0,1674	0,0403
Yahoo! Movies	**0,1013**	**0,01348**	**0,0021**

Observing the results from Table 6.3, Att-Item-KNN performs well in terms of adherence since, in approximately 75% of cases, even considering only the main feature of the explanation, it falls into the set of the item genres, as for Movielens dataset family. This performance is a 10% lower for Yahoo! Movies. In contrast with this result, VSM shows poor performances on both dataset of the Movielens family, by failing half the time about Movielens 1M as regards adherence. A surprising result is achieved for Yahoo! Movies dataset because, enlarging the study to the first three features among the explanation, the error is almost completely absent. The reasons we found to explain this difference in the performances concern the characteristics and the quality of the dataset, as we highlight later on.

6.4.2 Discussion

This work investigates how well a *post-hoc* approach based on local surrogates – such as the LIME-RS algorithm – explains a recommendation. Instead of studying the impact of explanations on users (that is a well-studied topic in the literature and is beyond our scope), we focus on objective evidences that could emerge. In this respect, we have designed specific experiments, which introduced two different metrics, to evaluate adherence and constancy for this kind of algorithms. For instance, Table 6.2 shows a different behavior for Att-Item-kNN and VSM. On the one hand, Att-Item-kNN seems to guarantee a good constancy in explanations up to the third feature. This suggests that an explanation that exploits the first three features of the list produced by LIME-RS could be barely considered as reliable (i.e., reaching a constancy of 0.69 on Movielens

1M). On the other hand, VSM exhibits a much more "stable" behavior, demonstrating in all cases (except for the first feature with Movielens 1M) better performance than Att-Item-kNN in terms of constancy, with peaks up to 97%. A straightforward consequence of these observations could be analyzed in terms of confidence or probability. If the constancy steadily decreases, it means that the probability that LIME-RS suggests the same explanatory feature decreases. In practical terms, we could say that LIME-RS is less confident about its explanation. In fact, this is the behavior of Att-Item-kNN. Conversely, VSM shows high values of constancy, resulting in a more "deterministic" behavior. With VSM, LIME-RS is more confident of its explanations. This could increase user's trustworthiness, since LIME-RS behavior is more reliable.

However, these results could also be interpreted together with the ones from Table 6.3. They show how often at least one feature – out of k features provided by LIME-RS– adheres to the features that describe the item being explained. In other words, they measure the probability that LIME-RS succeeds in reconstructing at least one feature of a specific item. Combining the results of Table 6.2 and those of Table 6.3, Att-Item-kNN, as already mentioned, shows good performance regarding adherence and identifies 3 times out of 4 the first fundamental feature of the explanation among those present in the set of features originally associated with the item. As expected, if the number k of LIME-RS-reconstructed features increases, the number of times such a set has a nonempty intersection (with the features belonging to the item) – *i.e.,* adherence – increases. It could be noted that Att-Item-kNN on Yahoo! Movies shows the worst behavior in terms of adherence. VSM shows a different behavior. Despite the excellent performance regarding constancy, it could be observed that on both Movielens datasets, the performance in terms of adherence is poor, and worse for Movielens 1M than for Movielens Small. Surprisingly, on Yahoo! Movies, VSM performs much better, and the errors are almost negligible.

The difference between the two models could be due to many reasons. In the following we analyze possible relations between such behaviors and two of them: *popularity bias in the dataset* and *characteristics of side information*. On the one hand, if the dataset is affected by popularity bias, it would be a well-studied cause of confusion for LIME-RS. On the other hand, the characteristics of the side information associated with the datasets could dramatically influence the performance of the two recommendation models. To assess these hypotheses, we have evaluated (see Table 6.4) the recommendation lists produced by Att-Item-kNN and VSM considering nDCG, Hit Rate (HR), Mean Average Precision (MAP), and Mean Reciprocal Rank (MRR).

Table 6.4 shows that the chosen datasets are strongly affected by popularity bias.

Table 6.4 Results of the experiments on the models involved in the experiments. Models are optimized according to the value of nDCG.

model	nDCG	Recall	HR	Precision	MAP	MRR
Movielens 1m						
Random	0,0051	0,0028	0,0869	0,0098	0,0094	0,0264
MostPop	0,0845	0,0379	0,4548	0,104	0,115	0,2205
Att-Item-kNN	0,0229	0,0165	0,2425	0,0383	0,0387	0,0888
VSM	0,0173	0,0109	0,2106	0,0292	0,0306	0,0741
Movielens Small						
Random	0,0030	0,0013	0,0492	0,0049	0,0068	0,0205
MostPop	0,0715	0,0389	0,3902	0,0748	0,0912	0,1961
Att-Item-kNN	0,0124	0,0068	0,1459	0,0197	0,0191	0,0484
VSM	0,0085	0,0056	0,1000	0,0111	0,0123	0.0350
Yahoo! Movies						
Random	0,0005	0,0008	0,0051	0,0005	0,0005	0,0015
MostPop	0,2188	0,2589	0,596	0,1067	0,1501	0,3447
Att-Item-kNN	0,0215	0,0262	0,1198	0,0132	0,0155	0,0435
VSM	0,0131	0,0171	0,0754	0,0081	0,0092	0,0261

Indeed, MostPop is the best performing approach, and the two "personalized" models fail to produce accurate results. This triggers the second aspect that concerns the quality of the content. The results suggest that the side information is not good enough to boost the recommendation systems in producing meaningful recommendations. In fact, the three datasets seem to have an informative content that is not adequate to generate appealing recommendations. We observe that, from an informative point of view, the Yahoo! Movies dataset is slightly more complete: 22 genres against the 18 genres available on Movielens. Although the VSM model does not show excellent performance, in combination with LIME-RS, it provides explanations that are very reliable in terms of constancy (see Table 6.2) and adherence (see Table 6.3) to the actual content of the items being explained.

From the designer perspective, there is also a pragmatic way to look at the experimental results. Suppose a developer needs an off-the-shelf way of generating explanations for recommendations, and chooses LIME-RS to do that. Our results suggest that if the explainer employs a Movielens dataset with Att-Item-kNN model, then it is better to run the explainer several times. Indeed, the first feature obtained for the explanation could change around 1 time every 5 trials (first column of Table 6.2), and once such a feature is obtained, it is better to check whether this feature is really among the ones describing the item, since 1 time out of 4 the feature can be wrong

(first column of Table 6.3). Moreover, if the explainer employs the Yahoo! Movies dataset with VSM model, then probably there is no need to run the explainer twice, since its behavior is constant 97% of the times, while the feature is wrong only 10% of the times. However, the low performance of such a model is to be taken into account.

6.5 Beyond post-hoc approach

The details that emerged in the previous section highlight how post-hoc approaches based on local surrogate models may prove ineffective. The explanations to the recommendations generated by such an approach may not be calibrated to the user profile, and one would run the risk of having unreliable justifications.

To make the explanations more effective for the user, several solutions have been proposed that are more likely to meet user-experience criteria. Moreover, some recent studies [180] straddling XAI and cognitive science have proven that causal explanations (particularly contrastive and counterfactual ones) are the most accommodating when it comes to providing an explanation to a generic process.

In order to introduce an approach that is capable of providing this kind of explanation, we will focus this proposal on a recommendation paradigm that takes into account not only the end user (consumer) but also the provider of the good/service being recommended. This framework is named Multi Stakeholder RSs (MS-RSs). Such systems are useful in a real recommendation scenario like e-commerce, where also the provider of products is involved in the recommendation process. Another classical scenario is dating, in which the recommendation has to be acceptable to both kinds of users of the transaction [272]. Following this idea, group RSs were proposed, with the aim of maximising the utility of each stakeholder in the group [174]. In this direction, it is clear that the MS-RSs approach is to devise a strategy that includes the utility of different stakeholders (like in a multi-side approach) and this approach was generalised to every recommendation task [3, 4]. Burke et al. [48] propose a general model for MS-RSs, which considers three kinds of users in the loop: the consumer who receives the recommendation, the system that supports the recommendation process, and the provider who feeds the system catalogue. Naturally, in the MS-RSs scenario, all involved kinds of users must be taken into account in the explanation process.

6.5.1 Notation

In this section, we formally define the viewpoint of each stakeholder, in terms of both her profile and the recommendation the MS-RSs gives her. This will set our notation for a formalization of counterfactual explanations of such recommendations in Sections 6.5.2 and 6.5.3.

In the present study, we envisage two types of stakeholders: consumers and providers (we leave the inclusion of the MS-RSs utility for future work). We denote the set of all consumers as $\mathcal{C} = \{c_1, c_2, \ldots\}$, and the set of providers as $\mathcal{P} = \{p_1, p_2, \ldots\}$.

Items are enumerated into a set $\mathcal{I} = \{I_1, I_2, \ldots\}$. To simplify the following formulas, we represent an item just by its index in $\mathcal{I}$, so that a list of items $\langle I_3, I_7, I_2\rangle$ (*e.g.,* a recommendation) will be just a list of natural numbers $\langle 3, 7, 2\rangle$. We do not delve in this section into the characteristics of items—*i.e.,* their features.

In general, the MS-RSs keeps a *profile* for each consumer c that collects her preferences or requirements. In this section, we consider a consumer profile as a list of items in decreasing consumer preference order: $P_c = \langle i_1, i_2, \ldots\rangle$—*i.e.,* an ordered list of items the consumer has chosen (or preferred in the past) the most. The recommendation process consists of a utility function $u_c : \mathcal{C} \times \mathcal{I} \to \mathbb{R}^+$. Such a utility can be represented by an accuracy, diversity, serendipity metric, or any other consumer utility, with the constraint that u_c is such that $u_c(c, i_1) \geq u_c(c, i_2) \geq u_c(c, i_3) \geq \ldots$, *i.e.,* the utility is coherent with the consumer's profile. A *recommendation for a consumer* c is an ordered list of items, denoted by $R_c = \langle i_1, i_2, \ldots\rangle$, meaning that the MS-RSs suggests the consumer the new item i_1 as most suitable, then item i_2 as a second choice, etc. The recommendation must be coherent with the consumer's utility, *i.e.,* $u_c(c, i_1) \geq u_c(c, i_2) \geq u_c(c, i_3) \geq \ldots$

Similarly, the *profile* of a provider p is a collection of her requirements. In this case, the provider's requirements represent some strategy that could maximize *e.g.,* profits, stock clearance, budget allocation, some other objectives, or a combination of some of them. Each strategy yields an ordered list of items, $P_p = \langle i_1, i_2, i_3 \ldots\rangle$, with the meaning that the provider would prefer to sell item i_1 the most, then item i_2, etc. Observe that such a set of strategies could be as large as needed, taking into account all possible choices the provider could make. Similarly to consumers, the recommender implements a utility function $u_p : \mathcal{P} \times \mathcal{I} \to \mathbb{R}^+$, giving a value to items from the provider's point of view, with the constraint that such utility is coherent with the provider's strategy, that is, $u_p(p, i_1) \geq u_p(p, i_2) \geq u_p(p, i_3) \geq \ldots$

Considering only the user utility in the recommendation task raises a problem called “Popularity Bias" [8] in which the RSs suggests the most popular items with

higher probability than less frequent ones. In this case, the problem was addressed by spreading diversity in the recommendation task [147]. Yet, approaches that promote diversity still lack the provider's perspective.

In a more recent work, Abdollahpouri et al. [2] propose another way to implement the recommendation task in a MS-RSs setting by using learning-to-reranking methodologies. The core problem is to compose the (sometimes) diverging interests of the two principal stakeholders: consumers and providers. Consumers want a personalized recommendation list that maximizes their utility, whereas providers want their products to have a higher probability of being sold. To find a new recommendation list which reflects a possible equilibrium point between consumer and provider utility functions a possible approach is to introduce a maximization problem of log-likelihood estimation. Following the same direction adopted by Abdollahpouri et al. [2] the problem becomes

$$\max_{\beta} \mathcal{L}(u_p|R_c,\mathcal{I}) = \sum_{j=1}^{m} \log(u_c(c,i_j)) + \beta \times \log(u_p(p,i_j))$$

In this formulation $\mathcal{L}$ denotes the loss of the log-likelihood estimation, m is the number of items presented in the recommendation list R_c. This maximization problem aims to fine-tune the parameter β to generate the new list of recommendation R_c^* optimized for both consumer and provider utility functions. Furthermore, the idea is to provide a new recommendation list that is not disruptive from the consumer's viewpoint. Hence, R_c^* is expected to be as similar as possible to R_c and this similarity could be expressed by a distance measure like the Kendall tau. This metric operates on the relative pairwise order of the items between the two lists to measure their difference.

Considering these two aspects, it is possible to introduce a new formulation for the generation of R_c^* in the form of

$$\min_{\beta,\gamma} \mathcal{L}(u_p|R_c,\mathcal{I}) = \mathcal{L}(u_p|\mathcal{I}) + \gamma(1 - \hat{K}(R_c, R_c^*))$$

The first term is referred to the optimization problem for generation R_c^* considering both consumer- and provider utilities. The term $\hat{K}(R_c, R_c^*)$ is the kernel-ized version [117] of the Kendall tau distance that regularized the loss as a similarity-based distance of R_c^* from the original R_c, while β are the weights of the optimized functions and γ is a hyper-parameter responsible for balancing the effect between the two terms of optimization.

6.5.2 Counterfactual Explanation

We now discuss counterfactual explanations in the context of MS-RSs. Counterfactual explanations follow the causality theory by Halpern et al. [102] for generating an explanation. Explanations depending on causality have not yet stood out in the RSs research area, but recently they are starting to attract interest.

We consider this kind of approach as the most suitable for MS-RSs, since we can distinguish between consumer-side and provider-side explanations, where each explanation does not reveal to a stakeholder the other stakeholders' preferences—as they are seen as exogenous causes.

Depending on the granularity of events, the computation of an explanation could change considerably. In the approach Verdeaux et al. [253], the events that cause a change in a consumer's recommendation list are purchases of single items; eliminating a suitable subset of such events would cause a rearrangement in the recommendation list, pushing lower items upwards. In that case, choosing a minimal set of purchases that change the consumer's preferences can be a computationally intractable problem [77]. However, for simplicity, in this study we treat the entire profile as an event, simplifying the search for a counterfactual cause of the recommendation to a simple rearrangement of the profile—in the simplest case, just a change in the first item. In this way, we decouple our analysis from computational problems, which we will deal with in future works. Explanations from the provider's perspective follow a similar approach: the endogenous cause of a particular recommendation to the consumer is the provider's strategy, that is, his profile as an ordered list of items. A counterfactual explanation looks for another strategy the provider could have chosen, which would have changed the recommendation.

More formally, a *counterfactual explanation* of a recommendation R_c^* *for a consumer* c, with profile P_c, is a pair $(P_c', R_c^{*'})$, where both $P_c \neq P_c'$ and $R_c^* \neq R_c^{*'}$, to be interpreted as follows: "Had Consumer c the profile P_c', the MS-RSs would recommend $R_c^{*'}$ instead of R_c^*".

In the simplest case, the recommender could focus on the first item of each list, providing an explanation of the following form: "I recommended you *Apple Phone XS* because based on your profile, you preferred *Samsung Galaxy S21* the most; if your most preferred item were *Samsung Galaxy S10*, I would suggest you *Google Pixel 5* instead".

On the provider's side, supposing the provider chose the strategy P_p, a *counterfactual explanation* of a recommendation R_c^* given to a consumer c, is a pair $(P_p', R_c^{*'})$, where

both $P_p \neq P'_p$ and $R^*_c \neq R^{*'}_c$, to be interpreted as follows: "Had provider p a different strategy P'_p, the MS-RSs would have recommended to c the new list $R^{*'}_c$".

Again, the simplest of such explanations would be to focus on one element only; for example: "I recommended to *Early adopter #1* the item *Google Pixel 5* because *Google Pixel 5* was the first one in your priority list; had you chosen Strategy P'_p, whose most prominent item is *Samsung Galaxy S10*, I would put this item in *Early adopter #1*'s recommendation".

Summarizing, counterfactual explanations never reveal to a stakeholder the other stakeholder's preferences, since they refer always to each stakeholder's own choices.

In their work, Halpern&Pearl identify two kinds of events, *exogenous* and *endogenous*. The former are determined by external factors and define the context. The latter are the factors an agent can change to influence a result and are in this way the expected causes of that result. In our MS-RSs scenario, we consider that events are exogenous or endogenous based on a stakeholder's perspective: namely, each stakeholder sees her actions as endogenous events, while all events corresponding to other stakeholder choices are exogenous.

Clearly, in a MS-RSs scenario, the only choices stakeholders can make are about their profile: a consumer might change her list of preferred items, while a provider might change his strategy. Consequently, we consider as events of our causal theory of counterfactuals the stakeholders' profiles.

6.5.3 Contrastive explanations.

Exploiting the formal models of causation by Halpern& Pearl and extending the causal chain definition provided by Hilton et al. [116], Miller [180] proposed contrastive explanations in the context of classical eXplainable Artificial Intelligence (XAI) for classification tasks. With the contrastive explanation, one wants to answer the question "Why P and not Q?". For example, a XAI system classifying pictures of animals should be able to justify its outcome by answering questions like, "Why did you classify that photo as a *spider* and not as a *crab*?" Of course, a contrastive explanation presumes that the user of the system already knows in some way the items to be contrasted.—in the previous example, the classes of spiders and crabs.

While this type of approach is claimed by Miller to be very effective in the context of XAI, when moving to the context of MS-RSs, however, it seems unsuitable because it may reveal indirectly other stakeholder's preferences. To make an example, suppose that a consumer already knows items *Apple Phone XS* and *Samsung Galaxy S21*, and suppose such items are completely equivalent from the consumer's perspective; yet the

MS-RSs recommended *Apple Phone XS* in a privileged position over *Samsung Galaxy S21*, just because this ordering meets the preferences of the provider. A contrastive explanation to the consumer question "Why did you put *Samsung Galaxy S21* so lower than *Apple Phone XS* if I like them both?" would have in this case no reason to put forward, but the provider's preferences. No possible answer to the consumer seems both adequate and trustworthy here. The provider's side contrastive explanations suffer from the same drawback: answering about the reasons of a big discrepancy in the recommendation of very similar—from the provider's preferences—items may reveal some consumer's preferences that she might have declared as private knowledge—information that MS-RSs is not authorized to reveal, adhering to EU GDPR, or other non-EU legislation.

6.6 Summary

In this first part of the chapter we shed a first light on the effectiveness of LIME-RS as a black-box explanation model in a recommendation scenario. We propose two different measures to understand how reliable an explanation based on LIME-RS is: *(i) constancy* was used to assess the impact of the random sampling phase of LIME-RS on the provided explanation – ideally the explanation should remain constant in spite of the sample used to obtain it; *(ii) adherence* was proposed to understand the reconstructive power of LIME-RS with respect to the features that belong to the item involved in the explanation – ideally, LIME-RS should provide an explanation that always adheres to the actual features of the recommended item.

To test both constancy and adherence, we trained and optimized two content-based recommendation models: Attribute Item-kNN (Att-Item-kNN), and a classical Vector Space Model. For each model, and for all datasets exploited in the study, we generated recommendation lists for all users. We exploited the first item of these top-10 lists to produce the explanations that were then the subject of our investigation. It turned out that for models built with a large collaborative input such as Att-Item-kNN, LIME-RS produces fairly constant explanations up to a length of three features. Moreover, these explanations turn out to be adherent with respect to the item between 65% and 75% of the cases in which only the first feature of the weighted vector of explanations is considered. VSM shows a different behavior where explanations are much more constant, but suffer a lot in terms of adherence, except for the Yahoo! Movies dataset for which the explanation model showed outstanding performance despite the poor ability of VSM to provide sound recommendations to users.

In our experiments, some evidence started to emerge highlighting that the adopted explanation model is conditioned not only by the accuracy of the black-box model it tries to explain but also by the quality of the side information used to train the model. The latter result deserves to be adequately investigated to search for a link at a higher level of detail. We plan to apply our experiments also to other recommendation models, to see whether the problems with adherence and constancy that we found for the two tested models show up also in other situations. We will also investigate what impact structured knowledge has on this performance by exploiting models capable of leveraging this type of content. In addition, it would also be the case to try different reference domains with richer datasets of side information to understand what impact content quality has on this type of explainer.

Finally, the second part of this chapter was devoted to formalizing a type of explanation derived from the causal approach, known in XAI as a counterfactual-type explanation. To contextualize these explanations, we motivated a Multi-Stakeholder recommendation scenario in which the different actors involved in the recommendation process are made explicit. In this scenario, each stakeholder provides the system with private information that may not be disclosed to other stakeholders. An explanation based on counterfactuals comes offbest since it can be based on the choices of each stakeholder without revealing the other's reserved information.

Chapter 7

Closing Remarks

An examination of the four themes that came up most often in our work—recommendation, reproducibility, beyond accuracy metrics, and explanation—was the first step in the process of writing this dissertation. In the end, we devised a synthesis of all of them and included an additional idea called personalization. The multitude of issues that served as the impetus for this three-year project not only provided us with the opportunity to investigate and suggest some possible answers but also made it possible for us to become aware of brand-new prospects.

In Chapter 3, we have first introduced Elliot, a new comprehensive framework RSs evaluation. The initial step that encouraged us to investigate the issue of reproducibility in RSs was the concept of having a tool that is able to manage an experimental pipeline and rigorously assess the performance of various recommendation models in a consistent manner. We compiled a list of the most effective procedures for each stage of the pipeline after being motivated to do so by the many works that have followed this line of inquiry over the course of the last several years. We combined them into a single software application that is able to provide both academics and practitioners with a straightforward user experience when they run an evaluation spanning different RSs models.

Inspired by Elliot's massive community-wide adoption, we set out to prove, on the one hand, how simple it is to reproduce results when the parameters of an experiment (dataset splitting, evaluation process, and metrics) are clear and well-defined. On the other hand, we built a comprehensive analysis of the many advanced models that belong to the CF family. The latter analysis demonstrated that the newest models are not always the most promising ones (in terms of accuracy), and that frequently the most historically established approaches still perform well if the testbed is the one that is built around best practices (dataset, splitting, and evaluation protocol),

which are widely utilized in the academic community. These discoveries are presented in Chapter 4, along with in-depth research into metrics that go beyond accuracy, which is added to the accuracy analysis. From this standpoint, we can see that the accuracy performance only occasionally weakens the performance of novelty, diversity, or bias. On the other hand, it is still of the opinion that it is impossible to produce a definitive ranking among all of these models to identify which is the finest.

Following this line of research regarding performance concerning beyond accuracy metrics, we shifted our focus to the latest recommendation approaches. In chapter 5, we introduce the family of RSs on a graph-based approach. They indeed represent one of the latest innovations introduced in this area to increase the level of personalization provided to users through recommendations. The investigation aimed to understand how this race toward greater accuracy combined novelty/diversity performance and Consumer and Provide Fairness (CP-fairness) issues. Based on this analysis and scrupulously analyzing the working mechanism of these models, we proposed revisiting classical message passing. This new formulation is the basis of the Edge Graph Collaborative Filtering (EDGCF) model presented in Section 5.6, which we have proven to be capable of generating accurate suggestions and providing low-polarized recommendations on the most popular items in the catalog.

Finally, building on the analysis of beyond-accuracy metrics and their importance in ensuring a better user experience for the final consumer of the suggestions, we addressed another crucial issue in the area of RSs (and one that turns out to be one of the pillars of Responsible AI): the generation of explanations to support the recommendations provided to the user. This area of interest has undergirded the entire research period of these three doctoral years and has been fundamental to my research period. We started from the assumption that to generate good explanations, there is a need for recommendations that are highly calibrated to the user, and how can we provide an explanation approach if the results of a recommendation model are not replicable? What sensible explanation can I provide to a user if I recommend a very popular item in his or her community of neighbors? These are the questions that prompted the outline work concerning this initial embryo of my research, which prompted me to open the horizon toward the issues dear to Responsible AI. The last chapter of this thesis analyzes the criticalities of a post-hoc explanation approach proposed for RSs. Through an experimental process, we showed how it is crucial to center explanations on the user preference model rather than the recommendation model. In addition, we proposed a formal model for generating counterfactual-type explanations for a particular class of RSs. This formal model embraces not only the idea of having a system with several

actors in a play to satisfy when it comes to the recommendation process, but it also adds an interesting perspective on handling sensitive information that should not be revealed during the explanation. This latter aspect must be addressed or investigated adequately in works dealing with this sphere of interest.

This is the goal that we have set out to accomplish, or more precisely, this is the goal that I have set out to accomplish with the help, support, collaboration, and encouragement of my collegues. I have made an attempt to contribute a little bit to the scientific community, but I am fully aware that my efforts are negligible in compared to the larger body of work. Nevertheless, I will continue to make these kinds of contributions. I am fully aware that no matter how much I study, develop, experiment, or write, I will never be able to completely eliminate the risk that I may make mistakes. I wouldn't have been able to learn anything new or improve the way I do things if I hadn't been subjected to things like rejection and failure in the past. Without the curiosity that pushes me and us forward, I doubt we would have accomplished the amazing things we have, nor would our researcher effort have been as powerful as it has been.

Bibliography

[1] Himan Abdollahpouri. "Popularity Bias in Ranking and Recommendation." In: *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society, AIES 2019, Honolulu, HI, USA, January 27-28, 2019*. Ed. by Vincent Conitzer, Gillian K. Hadfield, and Shannon Vallor. ACM, 2019, pp. 529–530. DOI: 10.1145/3306618.3314309.

[2] Himan Abdollahpouri, Gediminas Adomavicius, Robin Burke, Ido Guy, Dietmar Jannach, Toshihiro Kamishima, Jan Krasnodebski, and Luiz Augusto Pizzato. "Beyond Personalization: Research Directions in Multistakeholder Recommendation." In: (2019).

[3] Himan Abdollahpouri, Gediminas Adomavicius, Robin Burke, Ido Guy, Dietmar Jannach, Toshihiro Kamishima, Jan Krasnodebski, and Luiz Augusto Pizzato. "Multistakeholder recommendation: Survey and research directions." In: *User Model. User Adapt. Interact.* 30.1 (2020), pp. 127–158.

[4] Himan Abdollahpouri and Robin Burke. "Multi-stakeholder Recommendation and its Connection to Multi-sided Fairness." In: *RMSE@RecSys*. Vol. 2440. CEUR Workshop Proceedings. CEUR-WS.org, 2019.

[5] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. "Controlling Popularity Bias in Learning-to-Rank Recommendation." In: *Proceedings of the Eleventh ACM Conference on Recommender Systems, RecSys 2017, Como, Italy, August 27-31, 2017*. Ed. by Paolo Cremonesi, Francesco Ricci, Shlomo Berkovsky, and Alexander Tuzhilin. ACM, 2017, pp. 42–46. DOI: 10.1145/3109859.3109912.

[6] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. "Managing Popularity Bias in Recommender Systems with Personalized Re-Ranking." In: *Proceedings of the Thirty-Second International Florida Artificial Intelligence Research Society Conference, Sarasota, Florida, USA, May 19-22 2019*. Ed. by Roman Barták and Keith W. Brawner. AAAI Press, 2019, pp. 413–418.

[7] Himan Abdollahpouri, Mehdi Elahi, Masoud Mansoury, Shaghayegh Sahebi, Zahra Nazari, Allison Chaney, and Babak Loni. "MORS 2021: 1st Workshop on Multi-Objective Recommender Systems." In: *RecSys*. ACM, 2021, pp. 787–788.

[8] Himan Abdollahpouri, Masoud Mansoury, Robin Burke, and Bamshad Mobasher. "The Unfairness of Popularity Bias in Recommendation." In: *Proceedings of the Workshop on Recommendation in Multi-stakeholder Environments co-located with the 13th ACM Conference on Recommender Systems (RecSys 2019), Copenhagen, Denmark, September 20, 2019*. Ed. by Robin Burke, Himan Abdollahpouri, Edward C. Malthouse, K. P. Thai, and Yongfeng Zhang. Vol. 2440. CEUR Workshop Proceedings. CEUR-WS.org, 2019.

[9] Gediminas Adomavicius and Alexander Tuzhilin. "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions." In: *IEEE Trans. Knowl. Data Eng.* 17.6 (2005), pp. 734–749.

[10] David Alvarez-Melis and Tommi S. Jaakkola. "On the Robustness of Interpretability Methods." In: *CoRR* abs/1806.08049 (2018). arXiv: 1806.08049.

[11] David Alvarez-Melis and Tommi S. Jaakkola. "Towards Robust Interpretability with Self-Explaining Neural Networks." In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. Ed. by Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett. 2018, pp. 7786–7795.

[12] Vito Walter Anelli, Luca Belli, Yashar Deldjoo, Tommaso Di Noia, Antonio Ferrara, Fedelucio Narducci, and Claudio Pomo. "Pursuing Privacy in Recommender Systems: the View of Users and Researchers from Regulations to Applications." In: *RecSys*. ACM, 2021, pp. 838–841.

[13] Vito Walter Anelli, Alejandro Bellogín, Antonio Ferrara, Daniele Malitesta, Felice Antonio Merra, Claudio Pomo, Francesco M. Donini, Eugenio Di Sciascio, and Tommaso Di Noia. "The Challenging Reproducibility Task in Recommender Systems Research between Traditional and Deep Learning Models." In: *SEBD*. Vol. 3194. CEUR Workshop Proceedings. CEUR-WS.org, 2022, pp. 514–521.

[14] Vito Walter Anelli, Alejandro Bellogín, Antonio Ferrara, Daniele Malitesta, Felice Antonio Merra, Claudio Pomo, Francesco Maria Donini, and Tommaso Di Noia. "V-Elliot: Design, Evaluate and Tune Visual Recommender Systems." In: *RecSys*. ACM, 2021.

[15] Vito Walter Anelli, Alejandro Bellogín, Antonio Ferrara, Daniele Malitesta, Felice Antonio Merra, Claudio Pomo, Francesco Maria Donini, and Tommaso Di Noia. "Elliot: A Comprehensive and Rigorous Framework for Reproducible Recommender Systems Evaluation." In: *SIGIR*. ACM, 2021, pp. 2405–2414.

[16] Vito Walter Anelli, Alejandro Bellogín, Antonio Ferrara, Daniele Malitesta, Felice Antonio Merra, Claudio Pomo, Francesco Maria Donini, Eugenio Di Sciascio, and Tommaso Di Noia. "How to Perform Reproducible Experiments in the ELLIOT Recommendation Framework: Data Processing, Model Selection, and Performance Evaluation." In: *IIR*. Vol. 2947. CEUR Workshop Proceedings. CEUR-WS.org, 2021.

[17] Vito Walter Anelli, Alejandro Bellogín, Tommaso Di Noia, Francesco Maria Donini, Vincenzo Paparella, and Claudio Pomo. "Adherence and Constancy in LIME-RS Explanations for Recommendation (Long paper)." In: *KaRS/ComplexRec@RecSys*. Vol. 2960. CEUR Workshop Proceedings. CEUR-WS.org, 2021.

[18] Vito Walter Anelli, Alejandro Bellogín, Tommaso Di Noia, Francesco Maria Donini, Vincenzo Paparella, and Claudio Pomo. "An Analysis of Local Explanation with LIME-RS." In: *IIR*. Vol. 3177. CEUR Workshop Proceedings. CEUR-WS.org, 2022.

[19] Vito Walter Anelli, Alejandro Bellogín, Tommaso Di Noia, Dietmar Jannach, and Claudio Pomo. "Top-N Recommendation Algorithms: A Quest for the State-of-the-Art." In: *UMAP*. ACM, 2022, pp. 121–131.

[20] Vito Walter Anelli, Alejandro Bellogín, Tommaso Di Noia, and Claudio Pomo. "Reenvisioning the comparison between Neural Collaborative Filtering and Matrix Factorization." In: *RecSys*. ACM, 2021, pp. 521–529.

[21] Vito Walter Anelli, Yashar Deldjoo, Tommaso Di Noia, Eugenio Di Sciascio, Antonio Ferrara, Daniele Malitesta, and Claudio Pomo. "How Neighborhood Exploration influences Novelty and Diversity in Graph Collaborative Filtering." In: *MORS 2022: The Second Workshop on Multi-Objective Recommender Systems*. 2022.

[22] Vito Walter Anelli, Yashar Deldjoo, Tommaso Di Noia, Eugenio Di Sciascio, Antonio Ferrara, Daniele Malitesta, and Claudio Pomo. "Reshaping Graph Recommendation with Edge Graph Collaborative Filtering and Customer Reviews." In: *Workshop Deep Learning for Search and Recommendation*. 2022.

[23] Vito Walter Anelli, Tommaso Di Noia, Eugenio Di Sciascio, Azzurra Ragone, and Joseph Trotta. "Semantic Interpretation of Top-N Recommendations." In: *IEEE Transactions on Knowledge and Data Engineering* (2020), pp. 1–1. DOI: 10.1109/TKDE.2020.3010215.

[24] Vito Walter Anelli, Tommaso Di Noia, Eugenio Di Sciascio, Azzurra Ragone, and Joseph Trotta. "Semantic Interpretation of Top-N Recommendations." In: *IEEE Transactions on Knowledge and Data Engineering* (2020).

[25] Vito Walter Anelli, Tommaso Di Noia, Eugenio Di Sciascio, Claudio Pomo, and Azzurra Ragone. "On the discriminative power of hyper-parameters in cross-validation and how to choose them." In: *Proc. of the 13th ACM Conf. on Recommender Systems*. 2019, pp. 447–451.

[26] Vito Walter Anelli, Tommaso Di Noia, Eugenio Di Sciascio, Azzurra Ragone, and Joseph Trotta. "How to Make Latent Factors Interpretable by Feeding Factorization Machines with Knowledge Graphs." In: *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part I*. Ed. by Chiara Ghidini, Olaf Hartig, Maria Maleshkova, Vojtech Svátek, Isabel F. Cruz, Aidan Hogan, Jie Song, Maxime Lefrançois, and Fabien Gandon. Vol. 11778. Lecture Notes in Computer Science. Springer, 2019, pp. 38–56. DOI: 10.1007/978-3-030-30793-6_3.

[27] Vito Walter Anelli, Tommaso Di Noia, Eugenio Di Sciascio, Azzurra Ragone, and Joseph Trotta. "Local Popularity and Time in top-N Recommendation." In: *Advances in Information Retrieval - 41st European Conference on IR Research, ECIR 2019, Cologne, Germany, April 14-18, 2019, Proceedings, Part I*. Ed. by Leif Azzopardi, Benno Stein, Norbert Fuhr, Philipp Mayr, Claudia Hauff, and Djoerd Hiemstra. Vol. 11437. Lecture Notes in Computer Science. Springer, 2019, pp. 861–868. DOI: 10.1007/978-3-030-15712-8_63.

[28] Vito Walter Anelli et al. "RecSys 2020 Challenge Workshop: Engagement Prediction on Twitter's Home Timeline." In: *RecSys 2020: Fourteenth ACM Conference on Recommender Systems, Virtual Event, Brazil, September 22-26, 2020*. Ed. by Rodrygo L. T. Santos, Leandro Balby Marinho, Elizabeth M. Daly, Li Chen, Kim Falk, Noam Koenigstein, and Edleno Silva de Moura. ACM, 2020, pp. 623–627. DOI: 10.1145/3383313.3411532.

[29] Timothy G. Armstrong, Alistair Moffat, William Webber, and Justin Zobel. "Improvements That Don't Add Up: Ad-hoc Retrieval Results Since 1998." In: *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM '09)*. 2009, pp. 601–610.

[30] Ricardo Baeza-Yates. "Bias in Search and Recommender Systems." In: *RecSys*. ACM, 2020, p. 2.

[31] Vito Bellini, Giovanni Maria Biancofiore, Tommaso Di Noia, Eugenio Di Sciascio, Fedelucio Narducci, and Claudio Pomo. "GUapp: A Conversational Agent for Job Recommendation for the Italian Public Administration." In: *EAIS*. IEEE, 2020, pp. 1–7.

[32] Alejandro Bellogín, Iván Cantador, and Pablo Castells. "A study of heterogeneity in recommendations for a social music service." In: *HetRec@RecSys*. ACM, 2010, pp. 1–8.

[33] Alejandro Bellogín, Pablo Castells, and Iván Cantador. "Precision-oriented evaluation of recommender systems: an algorithmic comparison." In: *Proceedings of the 2011 ACM Conference on Recommender Systems, RecSys 2011, Chicago, IL, USA, October 23-27, 2011*. Ed. by Bamshad Mobasher, Robin D. Burke, Dietmar Jannach, and Gediminas Adomavicius. ACM, 2011, pp. 333–336. DOI: 10.1145/2043932.2043996.

[34] Alejandro Bellogín, Pablo Castells, and Iván Cantador. "Statistical biases in Information Retrieval metrics for recommender systems." In: *Inf. Retr. J.* 20.6 (2017), pp. 606–634. DOI: 10.1007/s10791-017-9312-z.

[35] Alejandro Bellogín and Alan Said. "Improving Accountability in Recommender Systems Research Through Reproducibility." In: *CoRR* abs/2102.00482 (2021). arXiv: 2102.00482.

[36] Alejandro Bellogín and Pablo Sánchez. "Revisiting Neighbourhood-Based Recommenders For Temporal Scenarios." In: *Proceedings of the 1st Workshop on Temporal Reasoning in Recommender Systems co-located with 11th International Conference on Recommender Systems (RecSys 2017), Como, Italy, August 27-31, 2017*. Ed. by Mária Bieliková, Veronika Bogina, Tsvi Kuflik, and Roy Sasson. Vol. 1922. CEUR Workshop Proceedings. CEUR-WS.org, 2017, pp. 40–44.

[37] James Bennett and Stan Lanning. "The netflix prize." In: *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, California, USA, August 12-15, 2007*. ACM, 2007.

[38] Rianne van den Berg, Thomas N. Kipf, and Max Welling. "Graph Convolutional Matrix Completion." In: *CoRR* abs/1706.02263 (2017).

[39] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. "Algorithms for Hyper-Parameter Optimization." In: *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*. Ed. by John Shawe-Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, and Kilian Q. Weinberger. 2011, pp. 2546–2554.

[40] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. "Algorithms for Hyper-Parameter Optimization." In: *Proceedings of the 24th International Conference on Neural Information Processing Systems*. NIPS'11. 2011, pp. 2546–2554.

[41] James Bergstra, Daniel Yamins, and David D. Cox. "Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures." In: *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*. Vol. 28. JMLR Workshop and Conference Proceedings. JMLR.org, 2013, pp. 115–123.

[42] Daniel Billsus and Michael J. Pazzani. "Learning Collaborative Information Filters." In: *Proceedings of the Fifteenth International Conference on Machine Learning*. 1998, pp. 46–54.

[43] Georgios Boltsis and Evaggelia Pitoura. "Bias disparity in graph-based collaborative filtering recommenders." In: *SAC*. ACM, 2022, pp. 1403–1409.

[44] Rodrigo Borges and Kostas Stefanidis. "On mitigating popularity bias in recommendations via variational autoencoders." In: *SAC*. ACM, 2021, pp. 1383–1389.

[45] Rodrigo Borges and Kostas Stefanidis. "On mitigating popularity bias in recommendations via variational autoencoders." In: *Proceedings of the 36th Annual ACM Symposium on Applied Computing*. 2021, pp. 1383–1389.

[46] Engin Bozdag. "Bias in Algorithmic Filtering and Personalization." In: *Ethics and Inf. Technol.* 15.3 (Sept. 2013), pp. 209–227. ISSN: 1388-1957. DOI: 10.1007/s10676-013-9321-6.

[47] Robin Burke, Nasim Sonboli, Masoud Mansoury, and Aldo Ordoñez-Gauger. "Balanced neighborhoods for fairness-aware collaborative recommendation." In: (2017).

[48] Robin D. Burke, Himan Abdollahpouri, Bamshad Mobasher, and Trinadh Gupta. "Towards Multi-Stakeholder Utility Evaluation of Recommender Systems." In: *UMAP (Extended Proceedings)*. Vol. 1618. CEUR Workshop Proceedings. CEUR-WS.org, 2016.

[49] Pedro G. Campos, Fernando Dıíez, and Iván Cantador. "Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols." In: *User Model. User Adapt. Interact.* 24.1-2 (2014), pp. 67–119. DOI: 10.1007/s11257-012-9136-x.

[50] Rocıío Cañamares and Pablo Castells. "On Target Item Sampling in Offline Recommender System Evaluation." In: *RecSys*. ACM, 2020, pp. 259–268.

[51] Pablo Castells, Neil J. Hurley, and Saul Vargas. "Novelty and Diversity in Recommender Systems." In: *Recommender Systems Handbook*. Ed. by Francesco Ricci, Lior Rokach, and Bracha Shapira. Springer, 2015, pp. 881–918. DOI: 10.1007/978-1-4899-7637-6_26.

[52] Dong-Kyu Chae, Jin-Soo Kang, Sang-Wook Kim, and Jung-Tae Lee. "CFGAN: A Generic Collaborative Filtering Framework based on Generative Adversarial Networks." In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*. Ed. by Alfredo Cuzzocrea et al. ACM, 2018, pp. 137–146. DOI: 10.1145/3269206.3271743.

[53] Zheng-Yi Chai, Ya-Lun Li, and Sifeng Zhu. "P-MOIA-RS: a multi-objective optimization and decision-making algorithm for recommendation systems." In: *J. Ambient Intell. Humaniz. Comput.* 12.1 (2021), pp. 443–454.

[54] Joymallya Chakraborty, Kewen Peng, and Tim Menzies. "Making Fair ML Software using Trustworthy Explanation." In: *35th IEEE/ACM International Conference on Automated Software Engineering, ASE 2020, Melbourne, Australia, September 21-25, 2020*. IEEE, 2020, pp. 1229–1233. DOI: 10.1145/3324884.3418932.

[55] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. "Measuring and Relieving the Over-Smoothing Problem for Graph Neural Networks from the Topological View." In: *AAAI*. AAAI Press, 2020, pp. 3438–3445.

[56] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. "Bias and Debias in Recommender System: A Survey and Future Directions." In: *CoRR* abs/2010.03240 (2020).

[57] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. "Attentive Collaborative Filtering: Multimedia Recommendation with Item- and Component-Level Attention." In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*. Ed. by Noriko Kando, Tetsuya Sakai, Hideo Joho, Hang Li, Arjen P. de Vries, and Ryen W. White. ACM, 2017, pp. 335–344. DOI: 10.1145/3077136.3080797.

[58] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. "Revisiting Graph Based Collaborative Filtering: A Linear Residual Graph Convolutional Network Approach." In: *AAAI*. AAAI Press, 2020, pp. 27–34.

[59] Xu Chen, Hanxiong Chen, Hongteng Xu, Yongfeng Zhang, Yixin Cao, Zheng Qin, and Hongyuan Zha. "Personalized Fashion Recommendation with Visual Explanations based on Multimodal Attention Network: Towards Visually Explainable Recommendation." In: *SIGIR*. ACM, 2019, pp. 765–774.

[60] Heng-Tze Cheng et al. "Wide & Deep Learning for Recommender Systems." In: *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, DLRS@RecSys 2016, Boston, MA, USA, September 15, 2016*. Ed. by Alexandros Karatzoglou, Balázs Hidasi, Domonkos Tikk, Oren Sar Shalom, Haggai Roitman, Bracha Shapira, and Lior Rokach. ACM, 2016, pp. 7–10. DOI: 10.1145/2988450.2988454.

[61] Evangelia Christakopoulou and George Karypis. "Local Item-Item Models For Top-N Recommendation." In: *Proceedings of the 10th ACM Conference on Recommender Systems*. 2016, pp. 67–74.

[62] Kenneth L. Clarkson. "Fast Algorithms for the All Nearest Neighbors Problem." In: *24th Annual Symposium on Foundations of Computer Science, Tucson, Arizona, USA, 7-9 November 1983*. IEEE Computer Society, 1983, pp. 226–232. DOI: 10.1109/SFCS.1983.16.

[63] Giandomenico Cornacchia, Francesco M. Donini, Fedelucio Narducci, Claudio Pomo, and Azzurra Ragone. "Explanation in Multi-Stakeholder Recommendation for Enterprise Decision Support Systems." In: *CAiSE Workshops*. Vol. 423. Lecture Notes in Business Information Processing. Springer, 2021, pp. 39–47.

[64] Paul Covington, Jay Adams, and Emre Sargin. "Deep Neural Networks for YouTube Recommendations." In: *RecSys*. ACM, 2016, pp. 191–198.

[65] Paolo Cremonesi and Dietmar Jannach. "Progress in recommender systems research: Crisis? What crisis?" In: *AI Magazine* 42.3 (2021), pp. 43–54.

[66] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. "Performance of recommender algorithms on top-n recommendation tasks." In: *RecSys*. ACM, 2010, pp. 39–46.

[67] George Cybenko. "Approximation by superpositions of a sigmoidal function." In: *Math. Control. Signals Syst.* 2.4 (1989), pp. 303–314.

[68] Maurizio Ferrari Dacrema, Simone Boglio, Paolo Cremonesi, and Dietmar Jannach. "A Troubling Analysis of Reproducibility and Progress in Recommender Systems Research." In: *ACM Trans. Inf. Syst.* 39.2 (2021), 20:1–20:49.

[69] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar 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 2019, Copenhagen, Denmark, September 16-20, 2019*. Ed. by Toine Bogers, Alan Said, Peter Brusilovsky, and Domonkos Tikk. ACM, 2019, pp. 101–109. DOI: 10.1145/3298689.3347058.

[70] Yashar Deldjoo, Vito Walter Anelli, Hamed Zamani, Alejandro Bellogin, and Tommaso Di Noia. "A flexible framework for evaluating user and item fairness in recommender systems." In: *User Modeling and User-Adapted Interaction* (2020), pp. 1–47.

[71] Yashar Deldjoo, Vito Walter Anelli, Hamed Zamani, Alejandro Bellogıín, and Tommaso Di Noia. "A flexible framework for evaluating user and item fairness in recommender systems." In: *User Model. User Adapt. Interact.* 31.3 (2021), pp. 457–511. DOI: 10.1007/s11257-020-09285-1.

[72] Yashar Deldjoo, Tommaso Di Noia, Daniele Malitesta, and Felice Antonio Merra. "Leveraging Content-Style Item Representation for Visual Recommendation." In: *ECIR (2)*. Vol. 13186. Lecture Notes in Computer Science. Springer, 2022, pp. 84–92.

[73] Joaquin Delgado and Naohiro Ishii. "Memory-based weighted majority prediction." In: *SIGIR Workshop Recomm. Syst. Citeseer*. Citeseer. 1999, p. 85.

[74] Mukund Deshpande and George Karypis. "Item-based top-N recommendation algorithms." In: *ACM Trans. Inf. Syst.* 22.1 (2004), pp. 143–177.

[75] Tommaso Di Noia, Francesco Maria Donini, Dietmar Jannach, Fedelucio Narducci, and Claudio Pomo. "Conversational recommendation: Theoretical model and complexity analysis." In: *Information Sciences* (2022). ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2022.07.169.

[76] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard S. Zemel. "Fairness Through Awareness." In: *CoRR* abs/1104.3913 (2011). arXiv: 1104.3913.

[77] Thomas Eiter and Thomas Lukasiewicz. "Complexity results for structure-based causality." In: *Artificial Intelligence* 142.1 (2002), pp. 53–89. ISSN: 0004-3702. DOI: https://doi.org/10.1016/S0004-3702(02)00271-0.

[78] Michael D. Ekstrand. "LensKit for Python: Next-Generation Software for Recommender Systems Experiments." In: *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*. Ed. by Mathieu d'Aquin, Stefan Dietze, Claudia Hauff, Edward Curry, and Philippe Cudré-Mauroux. ACM, 2020, pp. 2999–3006. DOI: 10.1145/3340531.3412778.

[79] Michael D. Ekstrand, Robin Burke, and Fernando Diaz. "Fairness and discrimination in recommendation and retrieval." In: *Proceedings of the 13th ACM Conference on Recommender Systems, RecSys 2019, Copenhagen, Denmark, September 16-20, 2019*. Ed. by Toine Bogers, Alan Said, Peter Brusilovsky, and Domonkos Tikk. ACM, 2019, pp. 576–577. DOI: 10.1145/3298689.3346964.

[80] Michael D. Ekstrand, Robin Burke, and Fernando 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 2019, Paris, France, July 21-25, 2019*. Ed. by Benjamin Piwowarski, Max Chevalier, Éric Gaussier, Yoelle Maarek, Jian-Yun Nie, and Falk Scholer. ACM, 2019, pp. 1403–1404. DOI: 10.1145/3331184.3331380.

[81] Michael D. Ekstrand, F. Maxwell Harper, Martijn C. Willemsen, and Joseph A. Konstan. "User perception of differences in recommender algorithms." In: *RecSys*. ACM, 2014, pp. 161–168.

[82] Michael D. Ekstrand, Michael Ludwig, Joseph A. Konstan, and John Riedl. "Rethinking the recommender research ecosystem: reproducibility, openness, and LensKit." In: *Proceedings of the 2011 ACM Conference on Recommender Systems, RecSys 2011, Chicago, IL, USA, October 23-27, 2011*. Ed. by Bamshad Mobasher, Robin D. Burke, Dietmar Jannach, and Gediminas Adomavicius. ACM, 2011, pp. 133–140. DOI: 10.1145/2043932.2043958.

[83] Michael D. Ekstrand, John Riedl, and Joseph A. Konstan. "Collaborative Filtering Recommender Systems." In: *Foundations and Trends in Human-Computer Interaction* 4.2 (2011), pp. 175–243.

[84] Ignacio Fernández-Tobıías, Iván Cantador, Paolo Tomeo, Vito Walter Anelli, and Tommaso Di Noia. "Addressing the user cold start with cross-domain collaborative filtering: exploiting item metadata in matrix factorization." In: *User Model. User Adapt. Interact.* 29.2 (2019), pp. 443–486. DOI: 10.1007/s11257-018-9217-6.

[85] Maurizio Ferrari Dacrema, Simone Boglio, Paolo Cremonesi, and Dietmar Jannach. "A Troubling Analysis of Reproducibility and Progress in Recommender Systems Research." In: *ACM Transactions on Information Systems (TOIS)* 39 (2 2021).

[86] François Fouss, Alain Pirotte, Jean-Michel Renders, and Marco Saerens. "Random-Walk Computation of Similarities between Nodes of a Graph with Application to Collaborative Recommendation." In: *IEEE Trans. Knowl. Data Eng.* 19.3 (2007), pp. 355–369.

[87] Ben Frederickson. *Fast python collaborative filtering for implicit datasets*. 2018.

[88] Zuohui Fu et al. "Fairness-Aware Explainable Recommendation over Knowledge Graphs." In: *SIGIR*. ACM, 2020, pp. 69–78.

[89] Simon Funk. *Netflix Update: Try This at Home*. 2006. URL: http://sifter.org/simon/journal/20061211.html.

[90] Simon Funk. *Netflix update: Try this at home*. 2006.

[91] Francesco Fusco, Michalis Vlachos, Vasileios Vasileiadis, Kathrin Wardatzky, and Johannes Schneider. "RecoNet: An Interpretable Neural Architecture for Recommender Systems." In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*. Ed. by Sarit Kraus. ijcai.org, 2019, pp. 2343–2349. DOI: 10.24963/ijcai.2019/325.

[92] Harold N. Gabow, Jon Louis Bentley, and Robert Endre Tarjan. "Scaling and Related Techniques for Geometry Problems." In: *Proceedings of the 16th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1984, Washington, DC, USA*. Ed. by Richard A. DeMillo. ACM, 1984, pp. 135–143. DOI: 10.1145/800057.808675.

[93] Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, and Lars Schmidt-Thieme. "Personalized Ranking for Non-Uniformly Sampled Items." In: *Proceedings of KDD Cup 2011 competition, San Diego, CA, USA, 2011*. Ed. by Gideon Dror, Yehuda Koren, and Markus Weimer. Vol. 18. JMLR Proceedings. JMLR.org, 2012, pp. 231–247.

[94] Zeno Gantner, Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. "MyMediaLite: a free recommender system library." In: *Proceedings of the 2011 ACM Conference on Recommender Systems, RecSys 2011, Chicago, IL, USA, October 23-27, 2011*. Ed. by Bamshad Mobasher, Robin D. Burke, Dietmar Jannach, and Gediminas Adomavicius. ACM, 2011, pp. 305–308. DOI: 10.1145/2043932.2043989.

[95] Jingyue Gao, Xiting Wang, Yasha Wang, and Xing Xie. "Explainable Recommendation through Attentive Multi-View Learning." In: *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 2019, pp. 3622–3629. DOI: 10.1609/aaai.v33i01.33013622.

[96] Fatih Gedikli, Dietmar Jannach, and Mouzhi Ge. "How should I explain? A comparison of different explanation types for recommender systems." In: *Int. J. Hum. Comput. Stud.* 72.4 (2014), pp. 367–382. DOI: 10.1016/j.ijhcs.2013.12.007.

[97] Marco de Gemmis, Pasquale Lops, Cataldo Musto, Fedelucio Narducci, and Giovanni Semeraro. "Semantics-Aware Content-Based Recommender Systems." In: *Recommender Systems Handbook*. Springer, 2015, pp. 119–159.

[98] Asela Gunawardana and Guy Shani. "Evaluating Recommender Systems." In: *Recommender Systems Handbook*. Springer, 2015, pp. 265–308.

[99] Guibing Guo, Jie Zhang, Zhu Sun, and Neil Yorke-Smith. "LibRec: A Java Library for Recommender Systems." In: *Posters, Demos, Late-breaking Results and Workshop Proceedings of the 23rd Conference on User Modeling, Adaptation, and Personalization (UMAP 2015), Dublin, Ireland, June 29 - July 3, 2015*. Ed. by Alexandra I. Cristea, Judith Masthoff, Alan Said, and Nava Tintarev. Vol. 1388. CEUR Workshop Proceedings. CEUR-WS.org, 2015.

[100] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. "DeepFM: A Factorization-Machine based Neural Network for CTR Prediction." In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*. Ed. by Carles Sierra. ijcai.org, 2017, pp. 1725–1731. DOI: 10.24963/ijcai.2017/239.

[101] Udit Gupta, Samuel Hsia, Vikram Saraph, Xiaodong Wang, Brandon Reagen, Gu-Yeon Wei, Hsien-Hsin S. Lee, David Brooks, and Carole-Jean Wu. "DeepRecSys: A System for Optimizing End-To-End At-Scale Neural Recommendation Inference." In: *47th ACM/IEEE Annual International Symposium on Computer Architecture, ISCA 2020, Valencia, Spain, May 30 - June 3, 2020*. IEEE, 2020, pp. 982–995. DOI: 10.1109/ISCA45697.2020.00084.

[102] Joseph Y. Halpern and Judea Pearl. "Causes and Explanations: A Structural-Model Approach - Part II: Explanations." In: *IJCAI*. Morgan Kaufmann, 2001, pp. 27–34.

[103] Christian Hansen, Rishabh Mehrotra, Casper Hansen, Brian Brost, Lucas Maystre, and Mounia Lalmas. "Shifting Consumption towards Diverse Content on Music Streaming Platforms." In: *WSDM*. ACM, 2021, pp. 238–246.

[104] F. Maxwell Harper and Joseph A. Konstan. "The MovieLens Datasets: History and Context." In: *ACM Trans. Interact. Intell. Syst.* 5.4 (2015).

[105] F. Maxwell Harper and Joseph A. Konstan. "The MovieLens Datasets: History and Context." In: *TiiS* 5.4 (2016), 19:1–19:19.

[106] Ruining He and Julian J. McAuley. "Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering." In: *WWW*. ACM, 2016, pp. 507–517.

[107] Ruining He and Julian J. McAuley. "VBPR: Visual Bayesian Personalized Ranking from Implicit Feedback." In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*. Ed. by Dale Schuurmans and Michael P. Wellman. AAAI Press, 2016, pp. 144–150.

[108] Xiangnan He and Tat-Seng Chua. "Neural Factorization Machines for Sparse Predictive Analytics." In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*. Ed. by Noriko Kando, Tetsuya Sakai, Hideo Joho, Hang Li, Arjen P. de Vries, and Ryen W. White. ACM, 2017, pp. 355–364. DOI: 10.1145/3077136.3080777.

[109] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong Zhang, and Meng Wang. "LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation." In: *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*. Ed. by Jimmy Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu. ACM, 2020, pp. 639–648. DOI: 10.1145/3397271.3401063.

[110] Xiangnan He, Xiaoyu Du, Xiang Wang, Feng Tian, Jinhui Tang, and Tat-Seng Chua. "Outer Product-based Neural Collaborative Filtering." In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*. Ed. by Jérôme Lang. ijcai.org, 2018, pp. 2227–2233. DOI: 10.24963/ijcai.2018/308.

[111] Xiangnan He, Zhankui He, Xiaoyu Du, and Tat-Seng Chua. "Adversarial Personalized Ranking for Recommendation." In: *SIGIR*. ACM, 2018, pp. 355–364.

[112] Xiangnan He, Zhankui He, Jingkuan Song, Zhenguang Liu, Yu-Gang Jiang, and Tat-Seng Chua. "NAIS: Neural Attentive Item Similarity Model for Recommendation." In: *IEEE Trans. Knowl. Data Eng.* 30.12 (2018), pp. 2354–2366. DOI: 10.1109/TKDE.2018.2831682.

[113] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. "Neural Collaborative Filtering." In: *WWW*. ACM, 2017, pp. 173–182.

[114] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. "Neural Collaborative Filtering." In: *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*. Ed. by Rick Barrett, Rick Cummings, Eugene Agichtein, and Evgeniy Gabrilovich. ACM, 2017, pp. 173–182. DOI: 10.1145/3038912.3052569.

[115] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. "Neural collaborative filtering." In: *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*. 2017, pp. 173–182.

[116] D. Hilton, J. McClure, and B. Slugoski. "The course of events: counterfactuals, causal sequences, and explanation." In: 2005.

[117] Thomas Hofmann, Bernhard Schölkopf, and Alexander J. Smola. "Kernel methods in machine learning." In: *Annals of Statistics* 36.3 (2008), pp. 1171–1220.

[118] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge J. Belongie, and Deborah Estrin. "Collaborative Metric Learning." In: *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*. Ed. by Rick Barrett, Rick Cummings, Eugene Agichtein, and Evgeniy Gabrilovich. ACM, 2017, pp. 193–201. DOI: 10.1145/3038912.3052639.

[119] Yifan Hu, Yehuda Koren, and Chris Volinsky. "Collaborative Filtering for Implicit Feedback Datasets." In: *ICDM*. IEEE Computer Society, 2008, pp. 263–272.

[120] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry P. Heck. "Learning deep structured semantic models for web search using clickthrough data." In: *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*. Ed. by Qi He, Arun Iyengar, Wolfgang Nejdl, Jian Pei, and Rajeev Rastogi. ACM, 2013, pp. 2333–2338. DOI: 10.1145/2505515.2505665.

[121] Nicolas Hug. "Surprise: A Python library for recommender systems." In: *J. Open Source Softw.* 5.52 (2020), p. 2174. DOI: 10.21105/joss.02174.

[122] Neil Hurley and Mi Zhang. "Novelty and Diversity in Top-N Recommendation - Analysis and Evaluation." In: *ACM Trans. Internet Techn.* 10.4 (2011), 14:1–14:30. DOI: 10.1145/1944339.1944341.

[123] Leo Iaquinta, Marco de Gemmis, Pasquale Lops, Giovanni Semeraro, Michele Filannino, and Piero Molino. "Introducing Serendipity in a Content-Based Recommender System." In: *HIS*. IEEE Computer Society, 2008, pp. 168–173.

[124] Mohsen Jamali and Martin Ester. "A matrix factorization technique with trust propagation for recommendation in social networks." In: *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010*. Ed. by Xavier Amatriain, Marc Torrens, Paul Resnick, and Markus Zanker. ACM, 2010, pp. 135–142. DOI: 10.1145/1864708.1864736.

[125] Dietmar Jannach, Lukas Lerche, Iman Kamehkhosh, and Michael Jugovac. "What recommenders recommend: an analysis of recommendation biases and possible countermeasures." In: *UMUAI* 25.5 (2015), pp. 427–491.

[126] Dietmar Jannach, Gabriel De Souza P. Moreira, and Even Oldridge. "Why Are Deep Learning Models Not Consistently Winning Recommender Systems Competitions Yet?" In: *ACM RecSys Challenge Workshop*. Online, 2020.

[127] Dietmar Jannach, Pearl Pu, Francesco Ricci, and Markus Zanker. "Recommender systems: Trends and frontiers." In: *AI Magazine* 43.2 (2022), pp. 145–150. DOI: https://doi.org/10.1002/aaai.12050. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/aaai.12050.

[128] Kalervo Järvelin and Jaana Kekäläinen. "Cumulated gain-based evaluation of IR techniques." In: *ACM Trans. Inf. Syst.* 20.4 (2002), pp. 422–446.

[129] Neil Jethani, Mukund Sudarshan, Ian Connick Covert, Su-In Lee, and Rajesh Ranganath. "FastSHAP: Real-Time Shapley Value Estimation." In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[130] Christopher C Johnson. "Logistic matrix factorization for implicit feedback data." In: *Advances in Neural Information Processing Systems* 27.78 (2014), pp. 1–9.

[131] Yu-Chin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. "Field-aware Factorization Machines for CTR Prediction." In: *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016*. Ed. by Shilad Sen, Werner Geyer, Jill Freyne, and Pablo Castells. ACM, 2016, pp. 43–50. DOI: 10.1145/2959100.2959134.

[132] Santosh Kabbur, Xia Ning, and George Karypis. "FISM: factored item similarity models for top-N recommender systems." In: *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*. Ed. by Inderjit S. Dhillon, Yehuda Koren, Rayid Ghani, Ted E. Senator, Paul Bradley, Rajesh Parekh, Jingrui He, Robert L. Grossman, and Ramasamy Uthurusamy. ACM, 2013, pp. 659–667. DOI: 10.1145/2487575.2487589.

[133] Marius Kaminskas and Derek Bridge. "Diversity, Serendipity, Novelty, and Coverage: A Survey and Empirical Analysis of Beyond-Accuracy Objectives in Recommender Systems." In: *ACM Trans. Interact. Intell. Syst.* 7.1 (2017), 2:1–2:42.

[134] Wang-Cheng Kang, Chen Fang, Zhaowen Wang, and Julian J. McAuley. "Visually-Aware Fashion Recommendation and Design with Generative Image Models." In: *ICDM*. IEEE Computer Society, 2017, pp. 207–216.

[135] Dong Hyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. "Convolutional Matrix Factorization for Document Context-Aware Recommendation." In: *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016*. Ed. by Shilad Sen, Werner Geyer, Jill Freyne, and Pablo Castells. ACM, 2016, pp. 233–240. DOI: 10.1145/2959100.2959165.

[136] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks." In: *ICLR (Poster)*. OpenReview.net, 2017.

[137] Joseph A. Konstan and Gediminas Adomavicius. "Toward identification and adoption of best practices in algorithmic recommender systems research." In: *Proceedings of the International Workshop on Reproducibility and Replication in Recommender Systems Evaluation, RepSys 2013, Hong Kong, China, October 12, 2013*. Ed. by Alejandro Bellogín, Pablo Castells, Alan Said, and Domonkos Tikk. ACM, 2013, pp. 23–28. DOI: 10.1145/2532508.2532513.

[138] Joseph A. Konstan and Loren G. Terveen. "Human-Centered Recommender Systems: Origins, Advances, Challenges, and Opportunities." In: *AI Mag.* 42.3 (2021), pp. 31–42.

[139] Yehuda Koren. "Factorization meets the neighborhood: a multifaceted collaborative filtering model." In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*. Ed. by Ying Li, Bing Liu, and Sunita Sarawagi. ACM, 2008, pp. 426–434. DOI: 10.1145/1401890.1401944.

[140] Yehuda Koren and Robert M. Bell. "Advances in Collaborative Filtering." In: *Recommender Systems Handbook*. Springer, 2015, pp. 77–118.

[141] Yehuda Koren, Robert M. Bell, and Chris Volinsky. "Matrix Factorization Techniques for Recommender Systems." In: *IEEE Computer* 42.8 (2009), pp. 30–37.

[142] Ralf Krestel, Peter Fankhauser, and Wolfgang Nejdl. "Latent dirichlet allocation for tag recommendation." In: *Proceedings of the 2009 ACM Conference on Recommender Systems, RecSys 2009, New York, NY, USA, October 23-25, 2009*. Ed. by Lawrence D. Bergman, Alexander Tuzhilin, Robin D. Burke, Alexander Felfernig, and Lars Schmidt-Thieme. ACM, 2009, pp. 61–68. DOI: 10.1145/1639714.1639726.

[143] Walid Krichene and Steffen Rendle. "On Sampled Metrics for Item Recommendation." In: *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*. Ed. by Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash. ACM, 2020, pp. 1748–1757.

[144] Walid Krichene and Steffen Rendle. "On Sampled Metrics for Item Recommendation." In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD '20. 2020, pp. 1748–1757.

[145] Maciej Kula. "Metadata Embeddings for User and Item Cold-start Recommendations." In: *CBRecSys@RecSys 2015*. 2015.

[146] Maciej Kula. *Spotlight*. https://github.com/maciejkula/spotlight. 2017.

[147] Matev Kunaver and Toma Porl. "Diversity in Recommender Systems A Survey." In: *Know.-Based Syst.* 123.C (May 2017), pp. 154–162.

[148] Mounia Lalmas. "Personalising and Diversifying the Listening Experience." In: *ICTIR*. ACM, 2020, p. 3.

[149] Alfonso Landin, Javier Parapar, and Álvaro Barreiro. "Novel and Diverse Recommendations by Leveraging Linear Models with User and Item Embeddings." In: *ECIR (2)*. Vol. 12036. Lecture Notes in Computer Science. Springer, 2020, pp. 215–222.

[150] Sara Latifi, Noemi Mauro, and Dietmar Jannach. "Session-aware Recommendation: A Surprising Quest for the State-of-the-art." In: *Information Sciences* 573 (2021), pp. 291–315.

[151] Daniel Lemire and Anna Maclachlan. "Slope One Predictors for Online Rating-Based Collaborative Filtering." In: *Proceedings of the 2005 SIAM International Conference on Data Mining, SDM 2005, Newport Beach, CA, USA, April 21-23, 2005*. Ed. by Hillol Kargupta, Jaideep Srivastava, Chandrika Kamath, and Arnold Goodman. SIAM, 2005, pp. 471–475. DOI: 10.1137/1.9781611972757.43.

[152] Mark Levy and Kris Jack. "Efficient top-n recommendation by linear regression." In: *RecSys Large Scale Recommender Systems Workshop*. 2013.

[153] Cheng-Te Li, Cheng Hsu, and Yang Zhang. "FairSR: Fairness-aware Sequential Recommendation through Multi-Task Learning with Preference Graph Embeddings." In: *ACM Trans. Intell. Syst. Technol.* 13.1 (2022), 16:1–16:21.

[154] Dong Li, Ruoming Jin, Jing Gao, and Zhi Liu. "On Sampling Top-K Recommendation Evaluation." In: *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*. Ed. by Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash. ACM, 2020, pp. 2114–2124.

[155] Yunqi Li, Hanxiong Chen, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. "User-oriented Fairness in Recommendation." In: *WWW*. ACM / IW3C2, 2021, pp. 624–632.

[156] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. "Variational Autoencoders for Collaborative Filtering." In: *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*. Ed. by Pierre-Antoine Champin, Fabien L. Gandon, Mounia Lalmas, and Panagiotis G. Ipeirotis. ACM, 2018, pp. 689–698. DOI: 10.1145/3178876.3186150.

[157] Jimmy Lin. "The neural hype and comparisons against weak baselines." In: *ACM SIGIR Forum* 52.2 (2019), pp. 40–51.

[158] Mingfeng Lin, Henry C. Lucas, and Galit Shmueli. "Research Commentary—Too Big to Fail: Large Samples and the p-Value Problem." In: *Information Systems Research* 24.4 (2013), pp. 906–917.

[159] Greg Linden, Brent Smith, and Jeremy York. "Amazon.com Recommendations: Item-to-Item Collaborative Filtering." In: *IEEE Internet Comput.* 7.1 (2003), pp. 76–80.

[160] Zachary C. Lipton. "The mythos of model interpretability." In: *Commun. ACM* 61.10 (2018), pp. 36–43. DOI: 10.1145/3233231.

[161] Qiang Liu, Shu Wu, and Liang Wang. "DeepStyle: Learning User Preferences for Visual Recommendation." In: *SIGIR*. ACM, 2017, pp. 841–844.

[162] Zhuoran Liu and Martha A. Larson. "Adversarial Item Promotion: Vulnerabilities at the Core of Top-N Recommenders that Use Images to Address Cold Start." In: *WWW*. ACM / IW3C2, 2021, pp. 3590–3602.

[163] Malte Ludewig, Noemi Mauro, Sara Latifi, and Dietmar Jannach. "Performance comparison of neural and non-neural approaches to session-based recommendation." In: *Proceedings of the 13th ACM Conference on Recommender Systems, RecSys 2019, Copenhagen, Denmark, September 16-20, 2019*. Ed. by Toine Bogers, Alan Said, Peter Brusilovsky, and Domonkos Tikk. ACM, 2019, pp. 462–466. DOI: 10.1145/3298689.3347041.

[164] Malte Ludewig, Noemi Mauro, Sara Latifi, and Dietmar Jannach. "Empirical analysis of session-based recommendation algorithms." In: *User Model. User Adapt. Interact.* 31.1 (2021), pp. 149–181. DOI: 10.1007/s11257-020-09277-1.

[165] Scott M. Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions." In: *NIPS*. 2017, pp. 4765–4774.

[166] Xin Luo, Mengchu Zhou, Yunni Xia, and Qingsheng Zhu. "An Efficient Non-Negative Matrix-Factorization-Based Approach to Collaborative Filtering for Recommender Systems." In: *IEEE Trans. Ind. Informatics* 10.2 (2014), pp. 1273–1284. DOI: 10.1109/TII.2014.2308433.

[167] Hao Ma, Haixuan Yang, Michael R. Lyu, and Irwin King. "SoRec: social recommendation using probabilistic matrix factorization." In: *Proceedings of the 17th ACM Conference on Information and Knowledge Management, CIKM 2008, Napa Valley, California, USA, October 26-30, 2008*. Ed. by James G. Shanahan, Sihem Amer-Yahia, Ioana Manolescu, Yi Zhang, David A. Evans, Aleksander Kolcz, Key-Sun Choi, and Abdur Chowdhury. ACM, 2008, pp. 931–940. DOI: 10.1145/1458082.1458205.

[168] Hao Ma, Dengyong Zhou, Chao Liu, Michael R. Lyu, and Irwin King. "Recommender systems with social regularization." In: *Proceedings of the Forth International Conference on Web Search and Web Data Mining, WSDM 2011, Hong Kong, China, February 9-12, 2011*. Ed. by Irwin King, Wolfgang Nejdl, and Hang Li. ACM, 2011, pp. 287–296. DOI: 10.1145/1935826.1935877.

[169] Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu. "Disentangled Graph Convolutional Networks." In: *ICML*. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 4212–4221.

[170] Masoud Mansoury, Himan Abdollahpouri, Mykola Pechenizkiy, Bamshad Mobasher, and Robin Burke. "FairMatch: A Graph-based Approach for Improving Aggregate Diversity in Recommender Systems." In: *UMAP*. ACM, 2020, pp. 154–162.

[171] Masoud Mansoury, Himan Abdollahpouri, Mykola Pechenizkiy, Bamshad Mobasher, and Robin Burke. "A Graph-Based Approach for Mitigating Multi-Sided Exposure Bias in Recommender Systems." In: *ACM Trans. Inf. Syst.* 40.2 (2022), 32:1–32:31.

[172] Masoud Mansoury, Bamshad Mobasher, Robin Burke, and Mykola Pechenizkiy. "Bias Disparity in Collaborative Recommendation: Algorithmic Evaluation and Comparison." In: *Proceedings of the Workshop on Recommendation in Multi-stakeholder Environments co-located with the 13th ACM Conference on Recommender Systems (RecSys 2019), Copenhagen, Denmark, September 20, 2019*. Ed. by Robin Burke, Himan Abdollahpouri, Edward C. Malthouse, K. P. Thai, and Yongfeng Zhang. Vol. 2440. CEUR Workshop Proceedings. CEUR-WS.org, 2019.

[173] Kelong Mao, Jieming Zhu, Xi Xiao, Biao Lu, Zhaowei Wang, and Xiuqiang He. "UltraGCN: Ultra Simplification of Graph Convolutional Networks for Recommendation." In: *CIKM*. ACM, 2021, pp. 1253–1262.

[174] Judith Masthoff. "Group Recommender Systems: Combining Individual Models." In: *Recommender Systems Handbook*. Springer, 2011, pp. 677–702.

[175] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. "Image-based recommendations on styles and substitutes." In: *Proc. of the 38th Int. ACM SIGIR Conf. on Research and Development in Inf. Retrieval*. 2015, pp. 43–52.

[176] Julian J. McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. "Image-Based Recommendations on Styles and Substitutes." In: *SIGIR 2015*. 2015.

[177] Sean M. McNee, John Riedl, and Joseph A. Konstan. "Being accurate is not enough: how accuracy metrics have hurt recommender systems." In: *Extended Abstracts Proceedings of the 2006 Conference on Human Factors in Computing Systems, CHI 2006, Montréal, Québec, Canada, April 22-27, 2006*. Ed. by Gary M. Olson and Robin Jeffries. ACM, 2006, pp. 1097–1101. DOI: 10.1145/1125451.1125659.

[178] Lei Meng, Fuli Feng, Xiangnan He, Xiaoyan Gao, and Tat-Seng Chua. "Heterogeneous Fusion of Semantic and Collaborative Information for Visually-Aware Food Recommendation." In: *ACM Multimedia*. ACM, 2020, pp. 3460–3468.

[179] Zaiqiao Meng, Richard McCreadie, Craig Macdonald, and Iadh Ounis. "Exploring Data Splitting Strategies for the Evaluation of Recommendation Models." In: *RecSys 2020: Fourteenth ACM Conference on Recommender Systems, Virtual Event, Brazil, September 22-26, 2020*. Ed. by Rodrygo L. T. Santos, Leandro Balby Marinho, Elizabeth M. Daly, Li Chen, Kim Falk, Noam Koenigstein, and Edleno Silva de Moura. ACM, 2020, pp. 681–686. DOI: 10.1145/3383313.3418479.

[180] Tim Miller. "Explanation in artificial intelligence: Insights from the social sciences." In: *Artif. Intell.* 267 (2019), pp. 1–38. DOI: 10.1016/j.artint.2018.07.007.

[181] Mohammadmehdi Naghiaei, Hossein A. Rahmani, and Yashar Deldjoo. "CPFair: Personalized Consumer and Producer Fairness Re-ranking for Recommender Systems." In: *SIGIR*. ACM, 2022, pp. 770–779.

[182] Jianmo Ni, Jiacheng Li, and Julian J. McAuley. "Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects." In: *EMNLP/IJCNLP (1)*. Association for Computational Linguistics, 2019, pp. 188–197.

[183] Xia Ning, Christian Desrosiers, and George Karypis. "A Comprehensive Survey of Neighborhood-Based Recommendation Methods." In: *Recommender Systems Handbook*. Springer, 2015, pp. 37–76.

[184] Xia Ning and George Karypis. "SLIM: Sparse Linear Methods for Top-N Recommender Systems." In: *11th IEEE International Conference on Data Mining, ICDM 2011, Vancouver, BC, Canada, December 11-14, 2011*. Ed. by Diane J. Cook, Jian Pei, Wei Wang, Osmar R. Zaı**********ane, and Xindong Wu. IEEE Computer Society, 2011, pp. 497–506. DOI: 10.1109/ICDM.2011.134.

[185] Wei Niu, James Caverlee, and Haokai Lu. "Neural Personalized Ranking for Image Recommendation." In: *WSDM 2018*. 2018.

[186] Caio Nóbrega and Leandro Balby Marinho. "Towards explaining recommendations through local surrogate models." In: *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, SAC 2019, Limassol, Cyprus, April 8-12, 2019*. Ed. by Chih-Cheng Hung and George A. Papadopoulos. ACM, 2019, pp. 1671–1678. DOI: 10.1145/3297280.3297443.

[187] Tommaso Di Noia, Roberto Mirizzi, Vito Claudio Ostuni, Davide Romito, and Markus Zanker. "Linked open data to support content-based recommender systems." In: *I-SEMANTICS 2012 - 8th International Conference on Semantic Systems, I-SEMANTICS '12, Graz, Austria, September 5-7, 2012*. Ed. by Valentina Presutti and Helena Sofia Pinto. ACM, 2012, pp. 1–8. DOI: 10.1145/2362499.2362501.

[188] Deng Pan, Xiangrui Li, Xin Li, and Dongxiao Zhu. "Explainable Recommendation via Interpretable Feature Mapping and Evaluation of Explainability." In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*. Ed. by Christian Bessiere. ijcai.org, 2020, pp. 2690–2696. DOI: 10.24963/ijcai.2020/373.

[189] Arkadiusz Paterek. "Improving regularized singular value decomposition for collaborative filtering." In: *Proceedings of KDD cup and workshop*. Vol. 2007. 2007, pp. 5–8.

[190] Bibek Paudel, Fabian Christoffel, Chris Newell, and Abraham Bernstein. "Updatable, Accurate, Diverse, and Scalable Recommendations for Interactive Applications." In: *ACM Trans. Interact. Intell. Syst.* 7.1 (2017), 1:1–1:34.

[191] Georgina Peake and Jun Wang. "Explanation Mining: Post Hoc Interpretability of Latent Factor Models for Recommendation Systems." In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*. Ed. by Yike Guo and Faisal Farooq. ACM, 2018, pp. 2060–2069. DOI: 10.1145/3219819.3220072.

[192] Shaowen Peng, Kazunari Sugiyama, and Tsunenori Mine. "SVD-GCN: A Simplified Graph Convolution Paradigm for Recommendation." In: *CoRR* abs/2208.12689 (2022).

[193] Gustavo Padilha Polleti, Hugo Neri Munhoz, and Fabio Gagliardi Cozman. "Explanations within conversational recommendation systems: improving coverage through knowledge graph embedding." In: *2020 AAAI Workshop on Interactive and Conversational Recommendation System. AAAI Press, New York City, New York, USA*. 2020.

[194] Tahleen A. Rahman, Bartlomiej Surma, Michael Backes, and Yang Zhang. "Fairwalk: Towards Fair Graph Embedding." In: *IJCAI*. ijcai.org, 2019, pp. 3289–3295.

[195] Navid Rekabsaz and Markus Schedl. "Do Neural Ranking Models Intensify Gender Bias?" In: *SIGIR*. ACM, 2020, pp. 2065–2068.

[196] Steffen Rendle. "Factorization Machines." In: *ICDM 2010, The 10th IEEE International Conference on Data Mining, Sydney, Australia, 14-17 December 2010*. 2010, pp. 995–1000. DOI: 10.1109/ICDM.2010.127.

[197] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. "BPR: Bayesian Personalized Ranking from Implicit Feedback." In: *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, June 18-21, 2009*. Ed. by JeffA. Bilmes and Andrew Y. Ng. AUAI Press, 2009, pp. 452–461.

[198] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. "Factorizing personalized Markov chains for next-basket recommendation." In: *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*. 2010, pp. 811–820. DOI: 10.1145/1772690.1772773.

[199] Steffen Rendle, Walid Krichene, Li Zhang, and John Anderson. "Neural Collaborative Filtering vs. Matrix Factorization Revisited." In: *Proceedings of the 14th ACM Conference on Recommender Systems (RecSys '20)*. 2020.

[200] Steffen Rendle, Walid Krichene, Li Zhang, and John R. Anderson. "Neural Collaborative Filtering vs. Matrix Factorization Revisited." In: *RecSys*. ACM, 2020, pp. 240–248.

[201] Steffen Rendle, Walid Krichene, Li Zhang, and John R. Anderson. "Neural Collaborative Filtering vs. Matrix Factorization Revisited." In: *RecSys 2020: Fourteenth ACM Conference on Recommender Systems, Virtual Event, Brazil, September 22-26, 2020*. Ed. by Rodrygo L. T. Santos, Leandro Balby Marinho, Elizabeth M. Daly, Li Chen, Kim Falk, Noam Koenigstein, and Edleno Silva de Moura. ACM, 2020, pp. 240–248. DOI: 10.1145/3383313.3412488.

[202] Steffen Rendle, Walid Krichene, Li Zhang, and Yehuda Koren. "iALS++: Speeding up Matrix Factorization with Subspace Optimization." In: *CoRR* abs/2110.14044 (2021). arXiv: 2110.14044.

[203] Steffen Rendle, Walid Krichene, Li Zhang, and Yehuda Koren. "Revisiting the Performance of iALS on Item Recommendation Benchmarks." In: *CoRR* abs/2110.14037 (2021). arXiv: 2110.14037.

[204] Steffen Rendle and Lars Schmidt-Thieme. "Pairwise interaction tensor factorization for personalized tag recommendation." In: *Proceedings of the Third International Conference on Web Search and Web Data Mining, WSDM 2010, New York, NY, USA, February 4-6, 2010*. 2010, pp. 81–90. DOI: 10.1145/1718487.1718498.

[205] Steffen Rendle, Li Zhang, and Yehuda Koren. "On the Difficulty of Evaluating Baselines: A Study on Recommender Systems." In: *CoRR* abs/1905.01395 (2019). arXiv: 1905.01395.

[206] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. "GroupLens: An Open Architecture for Collaborative Filtering of Netnews." In: *CSCW '94, Proceedings of the Conference on Computer Supported Cooperative Work, Chapel Hill, NC, USA, October 22-26, 1994*. Ed. by John B. Smith, F. Donelson Smith, and Thomas W. Malone. ACM, 1994, pp. 175–186. DOI: 10.1145/192844.192905.

[207] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. "GroupLens: An Open Architecture for Collaborative Filtering of Netnews." In: *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*. CSCW '94. 1994, pp. 175–186.

[208] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why Should I Trust You?": Explaining the Predictions of Any Classifier." In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. Ed. by Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi. ACM, 2016, pp. 1135–1144. DOI: 10.1145/2939672.2939778.

[209] Francesco Ricci, Lior Rokach, and Bracha Shapira, eds. *Recommender Systems Handbook*. Springer, 2015. ISBN: 978-1-4899-7636-9. DOI: 10.1007/978-1-4899-7637-6.

[210] Francesco Ricci, Lior Rokach, and Bracha Shapira. "Recommender Systems: Introduction and Challenges." In: *Recommender Systems Handbook*. Springer, 2015, pp. 1–34.

[211] Matthew Richardson, Rakesh Agrawal, and Pedro M. Domingos. "Trust Management for the Semantic Web." In: *ISWC*. Vol. 2870. Lecture Notes in Computer Science. Springer, 2003, pp. 351–368.

[212] C. J. van Rijsbergen. *Information Retrieval*. Butterworth, 1979.

[213] Juri Di Rocco, Davide Di Ruscio, Claudio Di Sipio, Phuong Thanh Nguyen, and Claudio Pomo. "On the Need for a Body of Knowledge on Recommender Systems (Short paper)." In: *KaRS/ComplexRec@RecSys*. Vol. 2960. CEUR Workshop Proceedings. CEUR-WS.org, 2021.

[214] Alan Said and Alejandro Bellogín. "Comparative recommender system evaluation: benchmarking recommendation frameworks." In: *Eighth ACM Conference on Recommender Systems, RecSys '14, Foster City, Silicon Valley, CA, USA - October 06 - 10, 2014*. Ed. by Alfred Kobsa, Michelle X. Zhou, Martin Ester, and Yehuda Koren. ACM, 2014, pp. 129–136. DOI: 10.1145/2645710.2645746.

[215] Alan Said and Alejandro Bellogín. "Rival: a toolkit to foster reproducibility in recommender system evaluation." In: *Eighth ACM Conference on Recommender Systems, RecSys '14, Foster City, Silicon Valley, CA, USA - October 06 - 10, 2014*. Ed. by Alfred Kobsa, Michelle X. Zhou, Martin Ester, and Yehuda Koren. ACM, 2014, pp. 371–372. DOI: 10.1145/2645710.2645712.

[216] Sean Saito, Eugene Chua, Nicholas Capel, and Rocco Hu. "Improving LIME Robustness with Smarter Locality Sampling." In: *CoRR* abs/2006.12302 (2020). arXiv: 2006.12302.

[217] Aghiles Salah, Quoc-Tuan Truong, and Hady W. Lauw. "Cornac: A Comparative Framework for Multimodal Recommender Systems." In: *J. Mach. Learn. Res.* 21 (2020), 95:1–95:5.

[218] Ruslan Salakhutdinov and Andriy Mnih. "Probabilistic Matrix Factorization." In: *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*. Ed. by John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis. Curran Associates, Inc., 2007, pp. 1257–1264.

[219] Ruslan Salakhutdinov and Andriy Mnih. "Bayesian probabilistic matrix factorization using Markov chain Monte Carlo." In: *ICML*. Vol. 307. ACM International Conference Proceeding Series. ACM, 2008, pp. 880–887.

[220] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. "Item-Based Collaborative Filtering Recommendation Algorithms." In: *Proceedings of the 10th International Conference on World Wide Web*. WWW '01. 2001, pp. 285–295.

[221] Badrul Munir Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. "Item-based collaborative filtering recommendation algorithms." In: *Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001*. Ed. by Vincent Y. Shen, Nobuo Saito, Michael R. Lyu, and Mary Ellen Zurko. ACM, 2001, pp. 285–295. DOI: 10.1145/371920.372071.

[222] J. Ben Schafer, Joseph A. Konstan, and John Riedl. "Recommender systems in e-commerce." In: *Proceedings of the First ACM Conference on Electronic Commerce (EC-99), Denver, CO, USA, November 3-5, 1999*. Ed. by Stuart I. Feldman and Michael P. Wellman. ACM, 1999, pp. 158–166. ISBN: 1-58113-176-3. DOI: 10.1145/336992.337035.

[223] Gunnar Schröder, Maik Thiele, and Wolfgang Lehner. "Setting Goals and Choosing Metrics for Recommender System Evaluations." In: *UCERSTI2 workshop at the 5th ACM conference on recommender systems, Chicago, USA*. Vol. 23. 2011, p. 53.

[224] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. "AutoRec: Autoencoders Meet Collaborative Filtering." In: *Proceedings of the 24th International Conference on World Wide Web Companion, WWW 2015, Florence, Italy, May 18-22, 2015 - Companion Volume*. Ed. by Aldo Gangemi, Stefano Leonardi, and Alessandro Panconesi. ACM, 2015, pp. 111–112. DOI: 10.1145/2740908.2742726.

[225] Guy Shani and Asela Gunawardana. "Evaluating Recommendation Systems." In: *Recommender Systems Handbook*. Springer, 2011, pp. 257–297.

[226] Yifei Shen, Yongji Wu, Yao Zhang, Caihua Shan, Jun Zhang, Khaled B. Letaief, and Dongsheng Li. "How Powerful is Graph Convolution for Recommendation?" In: *CIKM*. ACM, 2021, pp. 1619–1629.

[227] Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. "Fooling LIME and SHAP: Adversarial Attacks on Post hoc Explanation Methods." In: *AIES '20: AAAI/ACM Conference on AI, Ethics, and Society, New York, NY, USA, February 7-8, 2020*. Ed. by Annette N. Markham, Julia Powles, Toby Walsh, and Anne L. Washington. ACM, 2020, pp. 180–186. DOI: 10.1145/3375627.3375830.

[228] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. "Session-Based Social Recommendation via Dynamic Graph Attention Networks." In: *WSDM*. ACM, 2019, pp. 555–563.

[229] Dusan Stamenkovic, Alexandros Karatzoglou, Ioannis Arapakis, Xin Xin, and Kleomenis Katevas. "Choosing the Best of Both Worlds: Diverse and Novel Recommendations through Multi-Objective Reinforcement Learning." In: *WSDM*. ACM, 2022, pp. 957–965.

[230] Harald Steck. "Evaluation of recommendations: rating-prediction and ranking." In: *RecSys*. ACM, 2013, pp. 213–220.

[231] Harald Steck. "Embarrassingly Shallow Autoencoders for Sparse Data." In: *WWW*. ACM, 2019, pp. 3251–3257.

[232] Harald Steck, Linas Baltrunas, Ehtsham Elahi, Dawen Liang, Yves Raimond, and Justin Basilico. "Deep Learning for Recommender Systems: A Netflix Case Study." In: *AI Magazine* 42.3 (2021), pp. 7–18.

[233] Erik Strumbelj and Igor Kononenko. "Explaining prediction models and individual predictions with feature contributions." In: *Knowl. Inf. Syst.* 41.3 (2014), pp. 647–665. DOI: 10.1007/s10115-013-0679-x.

[234] Jianing Sun, Zhaoyue Cheng, Saba Zuberi, Felipe Pérez, and Maksims Volkovs. "HGCF: Hyperbolic Graph Convolution Networks for Collaborative Filtering." In: *WWW*. ACM / IW3C2, 2021, pp. 593–601.

[235] Jianing Sun et al. "A Framework for Recommending Accurate and Diverse Items Using Bayesian Graph Convolutional Neural Networks." In: *KDD*. ACM, 2020, pp. 2030–2039.

[236] Zhu Sun, Di Yu, Hui Fang, Jie Yang, Xinghua Qu, Jie Zhang, and Cong Geng. "Are We Evaluating Rigorously? Benchmarking Recommendation for Reproducible Evaluation and Fair Comparison." In: *RecSys 2020: Fourteenth ACM Conference on Recommender Systems, Virtual Event, Brazil, September 22-26, 2020*. Ed. by Rodrygo L. T. Santos, Leandro Balby Marinho, Elizabeth M. Daly, Li Chen, Kim Falk, Noam Koenigstein, and Edleno Silva de Moura. ACM, 2020, pp. 23–32. DOI: 10.1145/3383313.3412489.

[237] Zhu Sun, Di Yu, Hui Fang, Jie Yang, Xinghua Qu, Jie Zhang, and Cong Geng. "Are We Evaluating Rigorously? Benchmarking Recommendation for Reproducible Evaluation and Fair Comparison." In: *Fourteenth ACM Conference on Recommender Systems*. 2020, pp. 23–32.

[238] Jiaxi Tang and Ke Wang. "Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding." In: *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018, Marina Del Rey, CA, USA, February 5-9, 2018*. Ed. by Yi Chang, Chengxiang Zhai, Yan Liu, and Yoelle Maarek. ACM, 2018, pp. 565–573. DOI: 10.1145/3159652.3159656.

[239] Jinhui Tang, Xiaoyu Du, Xiangnan He, Fajie Yuan, Qi Tian, and Tat-Seng Chua. "Adversarial Training Towards Robust Multimedia Recommender System." In: *IEEE Trans. Knowl. Data Eng.* 32.5 (2020), pp. 855–867.

[240] Yiyi Tao, Yiling Jia, Nan Wang, and Hongning Wang. "The FacT: Taming Latent Factor Models for Explainability with Factorization Trees." In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*. Ed. by Benjamin Piwowarski, Max Chevalier, Éric Gaussier, Yoelle Maarek, Jian-Yun Nie, and Falk Scholer. ACM, 2019, pp. 295–304. DOI: 10.1145/3331184.3331244.

[241] Zhulin Tao, Yinwei Wei, Xiang Wang, Xiangnan He, Xianglin Huang, and Tat-Seng Chua. "MGAT: Multimodal Graph Attention Network for Recommendation." In: *Inf. Process. Manag.* 57.5 (2020), p. 102277.

[242] Nava Tintarev and Judith Masthoff. "A Survey of Explanations in Recommender Systems." In: *ICDE Workshops*. IEEE Computer Society, 2007, pp. 801–810.

[243] Nava Tintarev and Judith Masthoff. "Explaining Recommendations: Design and Evaluation." In: *Recommender Systems Handbook*. Springer, 2015, pp. 353–382.

[244] Michael Tsang, Dehua Cheng, Hanpeng Liu, Xue Feng, Eric Zhou, and Yan Liu. "Feature Interaction Interpretability: A Case for Explaining Ad-Recommendation Systems via Neural Interaction Detection." In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[245] Virginia Tsintzou, Evaggelia Pitoura, and Panayiotis Tsaparas. "Bias Disparity in Recommendation Systems." In: *Proceedings of the Workshop on Recommendation in Multi-stakeholder Environments co-located with the 13th ACM Conference on Recommender Systems (RecSys 2019), Copenhagen, Denmark, September 20, 2019*. Ed. by Robin Burke, Himan Abdollahpouri, Edward C. Malthouse, K. P. Thai, and Yongfeng Zhang. Vol. 2440. CEUR Workshop Proceedings. CEUR-WS.org, 2019.

[246] Kosetsu Tsukuda and Masataka Goto. "DualDiv: diversifying items and explanation styles in explainable hybrid recommendation." In: *Proceedings of the 13th ACM Conference on Recommender Systems, RecSys 2019, Copenhagen, Denmark, September 16-20, 2019*. Ed. by Toine Bogers, Alan Said, Peter Brusilovsky, and Domonkos Tikk. ACM, 2019, pp. 398–402. DOI: 10.1145/3298689.3347063.

[247] Pravin M. Vaidya. "An O(n log n) Algorithm for the All-nearest.Neighbors Problem." In: *Discret. Comput. Geom.* 4 (1989), pp. 101–115. DOI: 10.1007/BF02187718.

[248] Daniel Valcarce, Alejandro Bellogín, Javier Parapar, and Pablo Castells. "On the robustness and discriminative power of information retrieval metrics for top-N recommendation." In: *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*. Ed. by Sole Pera, Michael D. Ekstrand, Xavier Amatriain, and John O'Donovan. ACM, 2018, pp. 260–268. DOI: 10.1145/3240323.3240347.

[249] Daniel Valcarce, Alejandro Bellogín, Javier Parapar, and Pablo Castells. "Assessing ranking metrics in top-N recommendation." In: *Inf. Retr. J.* 23.4 (2020), pp. 411–448.

[250] Saúl Vargas. "Novelty and diversity enhancement and evaluation in recommender systems and information retrieval." In: *SIGIR*. ACM, 2014, p. 1281.

[251] Saul Vargas and Pablo Castells. "Rank and relevance in novelty and diversity metrics for recommender systems." In: *RecSys*. 2011.

[252] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. "Graph Attention Networks." In: *ICLR (Poster)*. OpenReview.net, 2018.

[253] Willeme Verdeaux, Clément Moreau, Nicolas Labroche, and Patrick Marcel. "Causality based explanations in multi-stakeholder recommendations." In: *EDBT/ICDT Workshops*. Vol. 2578. CEUR Workshop Proceedings. CEUR-WS.org, 2020.

[254] Michael Matthias Voit and Heiko Paulheim. "Bias in Knowledge Graphs - An Empirical Study with Movie Recommendation and Different Language Editions of DBpedia." In: *LDK*. Vol. 93. OASIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 14:1–14:13.

[255] Ellen M. Voorhees. "The TREC-8 Question Answering Track Report." In: *TREC*. Vol. 500-246. NIST Special Publication. National Institute of Standards and Technology (NIST), 1999.

[256] Sandra Wachter, Brent Mittelstadt, and Chris Russell. "Counterfactual explanations without opening the black box: Automated decisions and the GDPR." In: *Harv. JL & Tech.* 31 (2017), p. 841.

[257] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. "IRGAN: A Minimax Game for Unifying Generative and Discriminative Information Retrieval Models." In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*. Ed. by Noriko Kando, Tetsuya Sakai, Hideo Joho, Hang Li, Arjen P. de Vries, and Ryen W. White. ACM, 2017, pp. 515–524. DOI: 10.1145/3077136.3080786.

[258] Nan Wang, Lu Lin, Jundong Li, and Hongning Wang. "Unbiased Graph Embedding with Biased Graph Observations." In: *WWW*. ACM, 2022, pp. 1423–1433.

[259] Shoujin Wang, Xiuzhen Zhang, Yan Wang, Huan Liu, and Francesco Ricci. "Trustworthy Recommender Systems." In: *CoRR* abs/2208.06265 (2022).

[260] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. "KGAT: Knowledge Graph Attention Network for Recommendation." In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*. Ed. by Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis. ACM, 2019, pp. 950–958. DOI: 10.1145/3292500.3330989.

[261] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. "Neural Graph Collaborative Filtering." In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*. Ed. by Benjamin Piwowarski, Max Chevalier, Éric Gaussier, Yoelle Maarek, Jian-Yun Nie, and Falk Scholer. ACM, 2019, pp. 165–174. DOI: 10.1145/3331184.3331267.

[262] Xiang Wang, Hongye Jin, An Zhang, Xiangnan He, Tong Xu, and Tat-Seng Chua. "Disentangled Graph Collaborative Filtering." In: *SIGIR*. ACM, 2020, pp. 1001–1010.

[263] Xiting Wang, Yiru Chen, Jie Yang, Le Wu, Zhengtao Wu, and Xing Xie. "A Reinforcement Learning Framework for Explainable Recommendation." In: *IEEE International Conference on Data Mining, ICDM 2018, Singapore, November 17-20, 2018*. IEEE Computer Society, 2018, pp. 587–596. DOI: 10.1109/ICDM.2018.00074.

[264] Yifan Wang, Suyao Tang, Yuntong Lei, Weiping Song, Sheng Wang, and Ming Zhang. "DisenHAN: Disentangled Heterogeneous Graph Attention Network for Recommendation." In: *CIKM*. ACM, 2020, pp. 1605–1614.

[265] Zhidan Wang, Wenwen Ye, Xu Chen, Wenqiang Zhang, Zhenlei Wang, Lixin Zou, and Weidong Liu. "Generative Session-based Recommendation." In: *WWW*. ACM, 2022, pp. 2227–2235.

[266] Chuhan Wu, Fangzhao Wu, Tao Qi, Suyu Ge, Yongfeng Huang, and Xing Xie. "Reviews Meet Graphs: Enhancing User and Item Representations for Recommendation with Hierarchical Attentive Graph Neural Network." In: *EMNLP/IJCNLP (1)*. Association for Computational Linguistics, 2019, pp. 4883–4892.

[267] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. "Self-supervised Graph Learning for Recommendation." In: *SIGIR*. ACM, 2021, pp. 726–735.

[268] Junkang Wu, Wentao Shi, Xuezhi Cao, Jiawei Chen, Wenqiang Lei, Fuzheng Zhang, Wei Wu, and Xiangnan He. "DisenKGAT: Knowledge Graph Embedding with Disentangled Graph Attention Network." In: *CIKM*. ACM, 2021, pp. 2140–2149.

[269] Le Wu, Lei Chen, Pengyang Shao, Richang Hong, Xiting Wang, and Meng Wang. "Learning Fair Representations for Recommendation: A Graph-based Perspective." In: *WWW*. ACM / IW3C2, 2021, pp. 2198–2208.

[270] Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. "Collaborative Denoising Auto-Encoders for Top-N Recommender Systems." In: *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, San Francisco, CA, USA, February 22-25, 2016*. Ed. by Paul N. Bennett, Vanja Josifovski, Jennifer Neville, and Filip Radlinski. ACM, 2016, pp. 153–162. DOI: 10.1145/2835776.2835837.

[271] Yuehong Wu and Juan Yang. "Dual Sequential Recommendation Integrating High-order Collaborative Relations via Graph Attention Networks." In: *IJCNN*. IEEE, 2021, pp. 1–8.

[272] Peng Xia, Benyuan Liu, Yizhou Sun, and Cindy X. Chen. "Reciprocal Recommendation System for Online Dating." In: *ASONAM*. ACM, 2015, pp. 234–241.

[273] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. "Attentional Factorization Machines: Learning the Weight of Feature Interactions via Attention Networks." In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*. Ed. by Carles Sierra. ijcai.org, 2017, pp. 3119–3125. DOI: 10.24963/ijcai.2017/435.

[274] Yongquan Xie, Zhengru Li, Tian Qin, Finn Tseng, Johannes Kristinsson, Shiqi Qiu, and Yi Lu Murphey. "Personalized Session-Based Recommendation Using Graph Attention Networks." In: *IJCNN*. IEEE, 2021, pp. 1–8.

[275] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Victor S. Sheng, Jiajie Xu, Fuzhen Zhuang, Junhua Fang, and Xiaofang Zhou. "Graph Contextualized Self-Attention Network for Session-based Recommendation." In: *IJCAI*. ijcai.org, 2019, pp. 3940–3946.

[276] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. "Deep Matrix Factorization Models for Recommender Systems." In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*. Ed. by Carles Sierra. ijcai.org, 2017, pp. 3203–3209. DOI: 10.24963/ijcai.2017/447.

[277] Longqi Yang, Eugene Bagdasaryan, Joshua Gruenstein, Cheng-Kang Hsieh, and Deborah Estrin. "OpenRec: A Modular Framework for Extensible and Adaptable Recommendation Algorithms." In: *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018, Marina Del Rey, CA, USA, February 5-9, 2018*. Ed. by Yi Chang, Chengxiang Zhai, Yan Liu, and Yoelle Maarek. ACM, 2018, pp. 664–672. DOI: 10.1145/3159652.3159681.

[278] Wenzhuo Yang, Jia Li, Chenxi Li, Latrice Barnett, Markus Anderle, Simo Arajärvi, Harshavardhan Utharavalli, Caiming Xiong, and Steven C. H. Hoi. "On the Diversity and Explainability of Recommender Systems: A Practical Framework for Enterprise App Recommendation." In: *CIKM*. ACM, 2021, pp. 4302–4311.

[279] Hongzhi Yin, Bin Cui, Jing Li, Junjie Yao, and Chen Chen. "Challenging the Long Tail Recommendation." In: *Proc. VLDB Endow.* 5.9 (2012), pp. 896–907. DOI: 10.14778/2311906.2311916.

[280] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. "Graph Convolutional Neural Networks for Web-Scale Recommender Systems." In: *KDD*. ACM, 2018, pp. 974–983.

[281] Junliang Yu, Min Gao, Hongzhi Yin, Jundong Li, Chongming Gao, and Qinyong Wang. "Generating Reliable Friends via Adversarial Training to Improve Social Recommendation." In: *2019 IEEE International Conference on Data Mining, ICDM 2019, Beijing, China, November 8-11, 2019*. Ed. by Jianyong Wang, Kyuseok Shim, and Xindong Wu. IEEE, 2019, pp. 768–777. DOI: 10.1109/ICDM.2019.00087.

[282] ChengXiang Zhai, William W. Cohen, and John D. Lafferty. "Beyond independent relevance: methods and evaluation metrics for subtopic retrieval." In: *SIGIR 2003: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, July 28 - August 1, 2003, Toronto, Canada*. Ed. by Charles L. A. Clarke, Gordon V. Cormack, Jamie Callan, David Hawking, and Alan F. Smeaton. ACM, 2003, pp. 10–17. DOI: 10.1145/860435.860440.

[283] Chuanyan Zhang and Xiaoguang Hong. "Challenging the Long Tail Recommendation on Heterogeneous Information Network." In: *ICDM (Workshops)*. IEEE, 2021, pp. 94–101.

[284] He Zhang, Bang Wu, Xingliang Yuan, Shirui Pan, Hanghang Tong, and Jian Pei. "Trustworthy Graph Neural Networks: Aspects, Methods and Trends." In: *CoRR* abs/2205.07424 (2022).

[285] Jinghao Zhang, Yanqiao Zhu, Qiang Liu, Shu Wu, Shuhui Wang, and Liang Wang. "Mining Latent Structures for Multimedia Recommendation." In: *ACM Multimedia*. ACM, 2021, pp. 3872–3880.

[286] Mengfei Zhang, Cheng Guo, Jiaqi Jin, Mao Pan, and Jinyun Fang. "Sequential Recommendation with Context-Aware Collaborative Graph Attention Networks." In: *IJCNN*. IEEE, 2021, pp. 1–8.

[287] Sheng Zhang, Weihong Wang, James Ford, and Fillia Makedon. "Learning from Incomplete Ratings Using Non-negative Matrix Factorization." In: *Proceedings of the Sixth SIAM International Conference on Data Mining, April 20-22, 2006, Bethesda, MD, USA*. Ed. by Joydeep Ghosh, Diane Lambert, David B. Skillicorn, and Jaideep Srivastava. SIAM, 2006, pp. 549–553. DOI: 10.1137/1.9781611972764.58.

[288] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. "Deep Learning Based Recommender System: A Survey and New Perspectives." In: *ACM Comput. Surv.* 52.1 (2019), 5:1–5:38. DOI: 10.1145/3285029.

[289] Yongfeng Zhang and Xu Chen. "Explainable Recommendation: A Survey and New Perspectives." In: *Found. Trends Inf. Retr.* 14.1 (2020), pp. 1–101. DOI: 10.1561/1500000066.

[290] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. "Explicit factor models for explainable recommendation based on phrase-level sentiment analysis." In: *SIGIR*. ACM, 2014, pp. 83–92.

[291] Minghao Zhao, Le Wu, Yile Liang, Lei Chen, Jian Zhang, Qilin Deng, Kai Wang, Xudong Shen, Tangjie Lv, and Runze Wu. "Investigating Accuracy-Novelty Performance for Graph-based Collaborative Filtering." In: *SIGIR*. ACM, 2022, pp. 50–59.

[292] Wayne Xin Zhao et al. "RecBole: Towards a Unified, Comprehensive and Efficient Framework for Recommendation Algorithms." In: *CoRR* abs/2011.01731 (2020). arXiv: 2011.01731.

[293] Yu Zheng, Chen Gao, Liang Chen, Depeng Jin, and Yong Li. "DGCN: Diversified Recommendation with Graph Convolutional Networks." In: *WWW*. ACM / IW3C2, 2021, pp. 401–412.

[294] Guorui Zhou, Xiaoqiang Zhu, Chengru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. "Deep Interest Network for Click-Through Rate Prediction." In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*. Ed. by Yike Guo and Faisal Farooq. ACM, 2018, pp. 1059–1068. DOI: 10.1145/3219819.3219823.

[295] Kaixiong Zhou, Xiao Huang, Yuening Li, Daochen Zha, Rui Chen, and Xia Hu. "Towards Deeper Graph Neural Networks with Differentiable Group Normalization." In: *NeurIPS*. 2020.

[296] Tao Zhou, Zoltán Kuscsik, Jian-Guo Liu, MatúšMedo, Joseph Rushton Wakeling, and Yi-Cheng Zhang. "Solving the apparent diversity-accuracy dilemma of recommender systems." In: *Proceedings of the National Academy of Sciences* 107.10 (2010), pp. 4511–4515.

[297] Yu Zhu, Jinghao Lin, Shibi He, Beidou Wang, Ziyu Guan, Haifeng Liu, and Deng Cai. "Addressing the Item Cold-Start Problem by Attribute-Driven Active Learning." In: *IEEE Trans. Knowl. Data Eng.* 32.4 (2020), pp. 631–644.

[298] Ziwei Zhu, Yun He, Xing Zhao, Yin Zhang, Jianling Wang, and James Caverlee. "Popularity-Opportunity Bias in Collaborative Filtering." In: *Proceedings of the Fourteenth ACM International Conference on Web Search and Data Mining (WSDM '21), March 8–12, 2021, Virtual Event, Israel*. ACM, 2021. DOI: https://doi.org/10.1145/3437963.3441820.

[299] Ziwei Zhu, Xia Hu, and James Caverlee. "Fairness-Aware Tensor-Based Recommendation." In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*. Ed. by Alfredo Cuzzocrea et al. ACM, 2018, pp. 1153–1162. DOI: 10.1145/3269206.3271795.

[300] Ziwei Zhu, Jianling Wang, and James Caverlee. "Measuring and Mitigating Item Under-Recommendation Bias in Personalized Ranking Systems." In: *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*. Ed. by Jimmy Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu. ACM, 2020, pp. 449–458. DOI: 10.1145/3397271.3401177.