Uncertainty in Recommender Systems

Victor Azevedo Coscrato
MSC
119222668

Thesis submitted for the degree of Doctor of Philosophy

NATIONAL UNIVERSITY OF IRELAND, CORK

College of Science, Engineering and Food Science

School of Computer Science & Information Technology
Center for Research Training in Artificial Intelligence

April 2024

Head of School: Professor Utz Roedig

Supervisor: Dr. Derek G. Bridge

Contents

List of Figures

List of Tables

I, Victor Azevedo Coscrato, certify that this thesis is my own work and has not been submitted for another degree at University College Cork or elsewhere.

Victor Azevedo Coscrato

Para Rafaela, Silmara e Marcos

Acknowledgements

This PhD was a long journey, filled with achievements. But it also was by far the most difficult phase of my life. On several occasions, I needed help in many different ways. To those who provided much-needed support so that I could complete this dissertation, I would like to express my sincere thanks.

First, to my supervisor, Derek Bridge, for all the support provided, which greatly exceeded his academic responsibilities. Derek's dedication to research is very impressive, and can be confirmed by his academic record. Of much more importance is how caring a supervisor Derek can be. In many cases, supervisory meetings must be less about research and a lot more about the researcher, and that I learned from him. I was truly lucky to have Derek as my supervisor.

Rafaela, my fiancee, deserves much special thanks. Rafaela and I have been together long before I started the PhD. As a result of my choice to move to Ireland, we had to live apart for a long time. That was until she moved to join me, beginning an adventure that I am so happy to remember. Among all those who kept me strong during the times of adversity, Rafaela deserves the most recognition. Sincerely, thank you.

Another special thanks to all my colleagues from UCC. The PhD journey is lonely at times, but I was always able to push the loneliness away thanks to the friends I made along the way. To all of my friendships that started at the research office, at the handball court, or anywhere in Cork, cheers.

A minha família, agradeço a todo o suporte que me amparou até onde estou, em especial minha mãe Silmara e meu pai Marcos. Essa conquista também é de vocês. Obrigado.

Finally, I am grateful to the CRT-AI for this opportunity, in particular to the program coordinator, Janet Choi, who strives to meet the needs of students. This thesis has emanated from research conducted with the financial support of Science Foundation Ireland under Grant No. 18/CRT/6223, which is co-funded under the European Regional Development Fund. I am grateful for all the individuals who had some form of assistance with any part of this work.

Victor Coscrato
Cork, April 2024

Abstract

Recommender Systems have emerged as a powerful tool in the information era. Due to the overwhelming number of items (products and services) currently offered on digital platforms, it is often necessary to use a system capable of ranking the items and offering those that are most relevant to each user. These systems typically use historical user-item interaction data to build models that can predict the relevance of each item to the user.

There has long been a focus on increasing recommendation accuracy through the development of new prediction models. However, this is just one of the ways to improve these systems. It is also possible to equip them with new tools that extend their functionality in different ways. The tools that we focus on in this dissertation are uncertainty estimators.

The problem of uncertainty is relevant to Recommender Systems in at least two ways: prediction uncertainty and label uncertainty. Prediction uncertainty is the expected imprecision of the predictions given by the system's model. Label uncertainty is the chance that interactions used to learn the prediction model are mislabeled. This dissertation reports by far the most extensive study of these two types of uncertainty, offering a varied set of methods for their estimation, ranging from heuristic data metrics to novel uncertainty prediction models.

In overview, this dissertation is the largest compilation of methods for estimating prediction uncertainty and label uncertainty in Recommender Systems to date. This collection includes already-existing methods – that we survey, rewrite in a common notation, implement, make available under an open license, and compare in-depth – and many original methods, some that derive directly from existing work, but others that involve complex modeling. We divide our work into three branches: prediction uncertainty in explicit feedback-based systems, prediction uncertainty in implicit feedback-based systems, and label uncertainty in implicit feedback systems.

While this dissertation proposes new uncertainty estimation methods, the novel work in this dissertation is not restricted to new estimation methods. We also propose new techniques for evaluating prediction uncertainty estimators. Furthermore, we present and validate novel ways of using uncertainty estimators to improve the operation of a Recommender System.

At the core of our research program, and for each of the three branches cited above, we have rigorous validation of our prediction and label uncertainty esti-

mation methods through large-scale, reproducible empirical studies on publicly available recommendation datasets that unveil important insights into the performance and usefulness of the proposed methods. These studies include both the novel and surveyed uncertainty estimation methods, and make use of the novel uncertainty evaluation techniques that we propose.

This work can be an important mechanism for promoting new research on this topic that is still largely unexplored in the world of Recommender Systems. Thus, this dissertation is a contribution to the field of Recommender Systems, not just in terms of an all-encompassing compendium of uncertainty estimation methods for practitioners, but also in guiding future work. Given that the landscape of Recommender Systems continues to evolve, our work is poised to shape the discourse about uncertainty in the field.

Chapter 1

Introduction

1.1 Background and Motivation

Advances in technology have made information, products and services easily accessible through digital platforms. Nowadays, numerous businesses offer a nearly uncountable number of news and media posts, products and services, collectively referred to as *items*. Some examples are: streaming services, such as Spotify and Netflix; marketplaces, such as Amazon; and social media websites, such as Facebook, Instagram and TikTok. Due to the huge volumes involved, it is difficult for platform users to discover items they would like to consume and to choose between candidate items.

Recommender Systems (RSs) have emerged as tools to assist item discovery in these massive catalogs. They work by suggesting items that might be appealing according to the user's consumption behaviour. A successful RS should help its users by exposing them to relevant items, especially ones that they may not have discovered for themselves, and it should help the business by increasing sales, revenue, customer retention and so forth.

The economic interest around RSs has led to a rapid growth in the research field. The majority of the research conducted has focused on improving recommendation accuracy by developing novel, and more sophisticated, recommendation algorithms. Currently, there is a plethora of recommendation algorithms being used in real-world problems, achieving very high levels of accuracy.

The main goal of a RS is to recommended items that are relevant to its users. Typically, this requires that its recommendations are *personalized*. That is, the

system must provide recommendations that are tailored to what it infers about each user's preferences. Nevertheless, there are other important considerations. It is often also desirable, for example, for a set of recommendations to be *diverse* and for at least some of its recommendations to be *serendipitous*. We leave the explanation of *diversity*, *serendipity* and other RS side goals to Chapter 2.

There remain many open research topics, and the one we explore in this dissertation is the estimation of uncertainty in RSs.

Uncertainty is common to every machine learning (ML) task [HV07, HW21]. In particular, model predictions carry a degree of uncertainty. This uncertainty is often neglected, but it can be beneficial if uncertainty is quantified and even exploited. The quantification of prediction uncertainty has different motivations in different domains. For example, in medical decision making, estimating the uncertainty of predictions is key to the management of risk [BBK19]. In weather forecasting, estimating uncertainty and presenting the estimates to users can increase credibility [WLY+19]. RSs are no exception to the rule. For reasons that we explain below, recommendations can be uncertain and, as we also explore briefly below, estimating their uncertainty can improve the operation of a RS.

The literature in the area of uncertainty in ML often categorizes uncertainty into two types: *aleatoric* and *epistemic* [HW21]. The first refers to the randomness of the data itself, which cannot be reduced by additional knowledge. The second type, which is the focus of this dissertation, instead refers to the uncertainty caused by lack of knowledge. Therefore, epistemic uncertainty reduces our ability to know which of several prediction models is the correct one, which ultimately leads to uncertainty around the chosen model's predictions. For this reason, we can define *prediction uncertainty* as the expected imprecision of a prediction. Some related work in the area uses terminology such as reliability [BGOZ18] or confidence [WLW+18, CTFLHT13], which, in the context of our work, we take to be the opposite of uncertainty, that is, the expected precision of a prediction.

In RSs, prediction uncertainty has many causes. First, to provide personalized recommendations, the system must infer the preferences of individual users. Naturally, each user interacts with the items in a particular way: they click, purchase, consume and/or rate in different ways. For example, some users interact with many items, while others interact with fewer items; some interact with very few items at all. Due to these particularities, the system becomes more knowledgeable about some users and less about others. This is one cause of uncertainty.

Similarly, each item has its own particular interaction pattern, which also affects the level of uncertainty in recommendations. For example, popular (frequently consumed) items tend to be fairly certain recommendations. On the other hand, a more niche item from the 'long-tail' is often a more uncertain recommendation.

Furthermore, in practical RSs, the number of items is typically very large, meaning that users will have interacted with only a tiny fraction of them. This is known as *sparsity*. It means the system must infer a user's preferences from a relatively small amount of information, leading sometimes to the generation of unsuccessful but also uncertain recommendations.

Finally, there are challenges in collecting reliable user-item interaction data [SAA14]. In the case of Explicit feedback-based Recommender Systems (ERSs) – the class of systems that rely on the explicit input of user ratings (explained more fully in Chapter 2) — ratings data may be unreliable due to its high degree of subjectivity [ZN09], which can be explained by variations in user behaviour, personality and mood; change of preferences over time; and also by different interpretations of the rating scales by the users [Blo98, APO09]. Moreover, ratings are usually collected from open systems, being subject to natural noise (e.g. typing errors) and even malicious noise (e.g. from shilling attacks) [OHS06]. Unreliable user-item interaction data may even be caused by accidents. This is particularly so in the case of Implicit feedback-based Recommender Systems (IRSs) – the class of systems that observe and record aspects of the user's interaction behaviour with the system, such as searches, clicks, downloads, purchases (again see Chapter 2). A mistaken click, for example, means that the interaction data is not wholly reliable.

In the case of IRSs, there is another – arguably much more problematic – form of data unreliability, which is the lack of 'negative' feedback. An IRS does not ask its users whether they liked or disliked an item they interacted with, or how much they liked/disliked it. These systems record simply the fact that the user interacted with the item. Therefore, there is no way to tell apart liked and disliked items, only observed and not observed interactions with items. The traditional way to overcome this challenge is to simply assume all items the user interacted with are relevant to them, and in turn items the user did not interact with are not relevant. This introduces another form of uncertainty in this type of system, that we denote *label uncertainty*. Label uncertainty is the chance that an interaction gets mislabeled, i.e. how likely it is that (i) an observed interaction is actually irrelevant or (ii) an unobserved interaction is actually relevant.

The presence of label uncertainty decreases the reliability of the input data that is used to learn the system. We leave further details for later (Chapter 5). For now, it is enough to understand that *label uncertainty* is another form of uncertainty and that it is rather different from *prediction uncertainty*.

Quantifying the uncertainty of an RS's predictions and recommendations, either in the form of *prediction uncertainty* or *label uncertainty*, is important [MLG+03, Maz13, CTFLHT13, WFZ+22]. Ideally, uncertainty estimates should help detect which predictions and recommendations are more or less likely to be wrong [BGOZ18, ZOBG18].

Focusing first on *prediction uncertainty*, its estimation can improve the operation of the RS in at least the following three ways.

First, estimates of prediction uncertainty can enhance the presentation of recommendations to users. This may make the RS more credible, increasing the user's trust and satisfaction. In the simplest case, we can present the degree of prediction uncertainty alongside each recommendation [MLG+03, CTFLHT13]. But there are other ways that uncertainty estimates could affect the presentation of a set of recommendations. For example, consider an RS that can offer explanations for its recommendations. The user is most likely to need explanations for recommendations where it is least clear why the item is being recommended. These are likely to be the recommendations about which the RS is least certain. The RS could use high uncertainty to trigger the offer of an explanation. This may avert the loss of credibility that can come from showing recommendations that do not make sense to the user.

Second, estimates of prediction uncertainty offer new strategies for selecting which items are suggested to the user [Maz13, OLCGPB21]. Conventionally, the recommended items are those that the model predicts to have highest relevance to the user. But, equipped with estimates of uncertainty, an RS might discard highly uncertain recommendations, even when their predicted relevance is high: the user is shown items that the RS is certain are relevant. On the other hand, a RS might deliberately include some items that it predicts as relevant but where the relevance is uncertain: from a user point-of-view, this coincides with the idea that RSs are tools for item discovery; from a system point-of-view, this helps the RS explore user tastes by seeking feedback on uncertain recommendations. An RS might even combine these strategies: playing-it-safe by selecting some recommendations that are certain even though they may be obvious to the user; but taking-a-risk by also recommending some of the less certain ones.

This is reminiscent of the exploration/exploitation trade-off that is widely recognized and studied in Reinforcement Learning and even in the RS literature, e.g. [BBG13, BU17]. In fact, controlling exploration/exploitation is arguably the application where uncertainty is most frequently used in RS [YB21, ZTS+17].

Third, these estimates may be useful internally to an RS. A concrete example is found in the CoRec system, which uses co-training [dCMC18]. CoRec comprises more than one model and the most certain predictions of one model are added to the training set of the other model. More generally, uncertainty measures might help a hybrid RS decide which of several models to use in a particular circumstance [Bur02a]. There are also examples of Active Learning in RSs in which users are prompted to rate items that the RS is uncertain about, e.g. [KM01].

Turning now to *label uncertainty*, its estimation may also improve a RS's operation, and in two main ways. First, similarly to prediction uncertainty, label uncertainty may also be used to forge new strategies for recommendation selection. For example, items which a user has not previously interacted with that present high label uncertainty can be deemed potentially relevant. This is because they were likely mislabeled as irrelevant. Therefore, we may modify the system to encourage the recommendation of such items.

Second, label uncertainty estimates may be an important tool to counter-act popularity bias [AMBM19] in RSs. Because unpopular items – or 'long-tail' items – have few interactions, they are frequently deemed irrelevant, yet this may not be correct. It is likely that some unpopular items would be relevant to a user. By estimating label uncertainty, a RS might find these unpopular mislabeled items and potentially recommend them to the user.

In the literature, there have been several attempts at estimating prediction uncertainty in RSs, and we explore them especially in Chapters 3 and 4 of this dissertation. However, we can identify several gaps in the field, deserving of more research. To begin with, most of the available prediction uncertainty estimation methods have not been compared to one another. An empirical comparison, with a standardized form of evaluation – which the literature also lacks – is crucial to assess the credibility of the different prediction uncertainty estimators. Another issue is the lack of clarity over their coverage. That is, there is no clear categorization of which systems each prediction uncertainty estimator is applicable to. For instance, some methods are model-agnostic, meaning that they can be applied to any RS, while others are useful only for a restricted class of systems.

Finally, many of these estimators are based on heuristics. We believe that a more sophisticated form of uncertainty modelling could lead to better uncertainty estimates.

Unlike prediction uncertainty, label uncertainty is a relatively new concept and has received very little attention in the RS literature. Therefore, our goal in this branch of our research is to enlarge the literature with well formulated modelling methods — see Chapter 5.

1.2 Contributions

This dissertation addresses prediction uncertainty and label uncertainty in recommender systems. More precisely, it discusses the various forms of uncertainty estimation in both explicit feedback-based RSs (Chapter 3) and implicit feedback-based RSs (Chapters 4 and 5). Moreover, it also explores how to properly evaluate such uncertainty estimates. Finally, it also explores use cases for uncertainty estimates. Below, we present the main contributions this dissertation makes to the field.

1.2.1 Uncertainty estimators for explicit feedback RSs

We conduct by far the most extensive aggregation of methods for prediction uncertainty estimation in Explicit feedback-based Recommender Systems (ERSs) (see Chapter 3). We define a taxonomy to classify the methods. Furthermore, we re-write all the methods using a common notation; for some of the methods, this was a considerable undertaking[1]. We are positive that this contribution will substantially facilitate future research in the field.

In addition, we also propose two new methods for prediction uncertainty estimation in ERSs, namely Ensemble (see Section 3.2.2) and CV-Bias (see Section 3.2.3). These new methods are variants of the methods we surveyed. According to many of our evaluation metrics, these variants are effective, performing better than the ones that inspired them [CB22]. In Table 1.1, we list all the uncertainty estimators for ERS that we compared, highlighting the novel ones introduced in this dissertation.

[1]It is instructive to compare, for example, our presentation of OrdRec in Section 3.2.5 with the original in [KS11] to see one instance of just how much work was involved in adopting our common notation.

Table 1.1: All the prediction uncertainty estimators for explicit feedback-based RSs that we compare. The ones that are original to this dissertation are highlighted in gray.

Uncertainty estimator	Introduced in	Defined in
NEG-ITEM-SUPPORT	[Maz13]	3.2.1
ITEM-VARIANCE	[AKK07]	3.2.1
RESAMPLE	[Maz13]	3.2.2
ENSEMBLE	Novel	3.2.2
EB-FunkSVD	[ZOBG18]	3.2.3
EB-Linear	Novel	3.2.3
CPMF	[WLW+18]	3.2.4
BeMF	[OLCGPB21]	3.2.5
OrdRec	[KS11]	3.2.5

1.2.2 Uncertainty estimators for implicit feedback RSs

We also conduct the largest aggregation of methods for uncertainty estimation in Implicit feedback-based Recommender Systems (IRSs). For IRSs, we distinguish between two forms of uncertainty: prediction uncertainty and label uncertainty. For this reason, we further divide our aggregation of the methods into two chapters.

Prediction uncertainty in IRSs

In Chapter 4, we explore methods for prediction uncertainty estimation in IRSs. The estimators in this chapter follow either from (i) heuristic data metrics or (ii) computing variability in the relevance estimator. In particular, we identify which uncertainty estimation methods from Chapter 3 are extensible to IRSs. Furthermore, we also explore methods for uncertainty quantification in Neural Networks (NNs).

Many high performing RSs in the literature use NNs. To the best of our knowledge, two of the methods for uncertainty quantification in Neural Networks (NNs), Bayesian Neural Networks and Deep Ensembles, have not been previously applied to recommender systems, and so we include these in our investigations. Therefore, none of the methods from Chapter 4 are original to this dissertation. What is novel is their application to IRSs. Table 1.2 lists all the prediction uncertainty estimators compared in the chapter.

Table 1.2: All the prediction uncertainty estimators for implicit feedback-based RSs that we compare in Chapter 4. We mark with * the methods that are to be found in the literature but, to the best of our knowledge, have never previously been applied to IRSs.

Uncertainty estimator	Introduced in	Defined in
NEG-USER-SUPPORT	[Maz13]*	4.2.1
NEG-ITEM-SUPPORT	[Maz13]*	4.2.1
ENSEMBLE	[Maz13]*	4.2.2
Bayesian Neural Networks	[BCKW15]*	4.2.2
Monte Carlo Dropout	[ZTS+17]	4.2.2
Deep Ensemble	[LPB17]*	4.2.2

Table 1.3: All the label uncertainty estimators for implicit feedback-based RSs that we explore in Chapter 5. The ones that are original to this dissertation are highlighted in gray.

Uncertainty estimator	Introduced in	Defined in
Aleatoric Uncertainty-aware Recommendation	[WFZ+22]	5.1
ICPMF	Novel	5.2.1
Gaussian-MLP	Novel	5.2.2
Gaussian Binary Ranking	Novel	5.3.2
Hierarchical Binary Ranking	Novel	5.3.3

Label uncertainty in IRSs

In Chapter 5, we discuss the learning of label uncertainty estimators in IRSs. Although Chapter 5, like Chapter 4, tackles the implicit feedback recommendation case, label uncertainty is very different from prediction uncertainty, both in motivation and in its use cases, which is why label uncertainty deserves its own chapter.

The main contribution of this chapter is the definition of four novel methods for label uncertainty estimation: ICPMF, Gaussian-MLP, Gaussian Binary Ranking (GBR) and Hierarchical Binary Ranking (HBR). This is a considerable undertaking considering that these models are quite different from each other. For instance, without jumping into details, Gaussian-MLP is neural-network based while HBR is a hierarchical Bayesian model. In addition, we extensively compare these methods against Aleatoric Uncertainty-aware Recommendation, the closest method from the literature. Table 1.3 lists all the uncertainty estimators compared in the chapter.

1.2.3 Evaluation of prediction uncertainty estimates in RS

We present an extensive review of methods for evaluating prediction uncertainty estimates in both explicit and implicit feedback-based RSs. In addition, we extend and improve the techniques and metrics for evaluating prediction uncertainty estimates in RSs. In particular, we analyse how prediction uncertainty correlates with dataset statistics; we fix possible issues with existing metrics; and we propose new forms of evaluation for prediction uncertainty estimates in the top-n recommendation task. As explained in contribution 1.2.5 below, using these extended and improved techniques and metrics, we are able to conduct rigorous and reproducible empirical comparisons in Chapters 3 to 5

1.2.4 Uses for uncertainty estimates in RSs

We review and expand on the methods to exploit prediction uncertainty estimates in both explicit and implicit feedback-based RSs. More specifically, for explicit feedback-based RS, we introduce the idea of uncertainty-aware ranking, drawing in part on a recommendation strategy proposed in [OLCGPB21]. In particular, in Chapter 3, we distinguish uncertainty-based filtering (UBF) and probability-of-relevance ranking (PRR). In Chapter 4, we extend the UBF strategy to IRSs.

Similarly, in Chapter 5, we explore Mislabeling-Aware Ranking (MAR), that derives from [WFZ+22], another form of uncertainty-aware ranking, but based instead on label uncertainty.

We show that, in some cases, it is possible to provide more accurate recommendations by making use of uncertainty estimates through UBF or PRR. We also show that, in some cases, MAR increases overall recommendation accuracy and, on top of that, also helps reduce popularity bias.

1.2.5 A reproducible empirical comparison of the uncertainty estimators

Prior to this work, the literature in the field was largely disconnected. Most of the methods that we survey have never been empirically compared to each other, and therefore there was no clarity on what the state-of-the-art was. Of course, none of them had been compared with the methods that are original to this dissertation. We perform the most comprehensive comparison of uncertainty

estimators for both ERSs and IRSs to date. In each case, we use popular and widely available datasets.

We provide implementations for all the methods in Tables 1.1, 1.2 and 1.3 under an open license. Therefore, all of our experiments are completely reproducible, and new developments in the field can benefit from our source code.

1.3 Notation

In this section, we introduce general notation — notation that will be used throughout this dissertation. Other notation, which is specific to particular models, will be introduced on an as-needed basis in the remainder of the dissertation. Table 1.4 summarizes the notation schematically.

We will assume that our data consists of a set of users U, a set of items I, and a set R of user-item interactions. When numeric ratings are available (explicit feedback), R is formed by user-item-rating triples, where triple $\langle u, i, r_{ui} \rangle \in R$ means that user u's rating of item i is $r_{ui} \in S$. S is the set of possible ratings – e.g. $\{1\star, \cdots, 5\star\}$. Otherwise, we assume we have unary implicit feedback and so R is a set of pairs $\langle u, i \rangle$, that contain the observed interactions. That is, if user u has interacted with item i, then $\langle u, i \rangle \in R$. In addition, in this setting, the set of unseen pairs (unobserved interactions) is represented by the symbol $R^- = \{\langle u, i \rangle | u \in U, i \in I, (u, i) \notin R\}$.

A user's profile $R_{u\cdot}$ is the set of items that user u has interacted with. Formally, $R_{u\cdot} = \{i : \langle u, i, r_{ui} \rangle \in R\}$ if the interactions are explicit ratings, or $R_{u\cdot} = \{i : \langle u, i \rangle \in R\}$ otherwise. Similarly, an item's support set $R_{\cdot i}$ is the set of users that interacted with item i: $R_{\cdot i} = \{u : \langle u, i, r_{ui} \rangle \in R\}$ in the case of explicit ratings, or $R_{\cdot i} = \{u : \langle u, i \rangle \in R\}$ for implicit feedback.

We will postpone a more extensive description of the remaining notation in the table (such as τ and ω) for later chapters.

1.4 Publications

The following publications were produced from the work in this dissertation:

- Victor Coscrato and Derek Bridge: *Estimating and Evaluating the Uncertainty of Rating Predictions and Top-n Recommendations in Recommender*

Table 1.4: The dissertation's notation. Notice that the same notation may have a slightly different definition for ERSs and IRSs.

General Notation	
U	The set of all users.
I	The set of all items.
F	A rating or relevance scoring function, $F : U \times I \to \mathbb{R}$. For conciseness, $F_{ui} := F(u, i)$.
G	An uncertainty prediction function, $G : U \times I \to \mathbb{R}$. For conciseness, $G_{ui} := G(u, i)$.
C_u	Candidate items, $C_u \subseteq I$, that might be recommended to user u.
Z_u^n	List of n items that are recommended to user u.
τ	The uncertainty threshold for Uncertainty-Based Filtering.
Explicit feedback-based Recommender Systems (ERSs)	
r_{ui}	User u's rating for item i.
S	The ordered set of m possible rating values, $S = \{s_1, \ldots, s_m\}$.
R	The set of all user-item-rating-triples.
$R_{u\cdot}$	User u's profile, i.e. $R_{u\cdot} = \{r_{ui} : \langle u, i, r_{ui}\rangle \in R\}$.
$R_{\cdot i}$	Item i's support set, i.e. $R_{\cdot i} = \{r_{ui} : \langle u, i, r_{ui}\rangle \in R\}$.
ω	Relevance threshold; item i is deemed relevant to user u if $r_{ui} \geq \omega$.
Implicit feedback-based Recommender Systems (IRSs)	
R	The set of all user-item interaction pairs.
R^-	The set of all unobserved user-item interactions, i.e. $R^- = \{\langle u, i\rangle \mid u \in U, i \in I, (u, i) \notin R\}$.
$R_{u\cdot}$	User u's profile, i.e. $R_{u\cdot} = \{r_{ui} : \langle u, i\rangle \in R\}$.
$R_{\cdot i}$	Item i's support set, i.e. $R_{\cdot i} = \{r_{ui} : \langle u, i\rangle \in R\}$.

Systems. ACM Trans. Recomm. Syst. 1, 2, Article 7 (June 2023), 34 pages. https://doi.org/10.1145/3584021

- Victor Coscrato and Derek Bridge: *Recommendation Uncertainty in Implicit Feedback Recommender Systems*. In: Longo, L., O'Reilly, R. (eds) Artificial Intelligence and Cognitive Science. AICS 2022. Communications in Computer and Information Science, vol 1662. Springer, Cham. https://doi.org/10.1007/978-3-031-26438-2_22

1.5 Outline of the Dissertation

The remainder of this dissertation is structured as follows.

In Chapter 2, we present a brief introduction to Recommender Systems. This chapter reviews, at a high level, the three stages of a recommender system: candidate selection, scoring and top-n recommendations. Rather than being an exhaustive review, the chapter focuses on models and evaluation methods that will be used in the chapters to follow. The chapter ends on a brief introduction to uncertainty in RSs, that sets the ground to the following three chapters.

In Chapters 3 to 5, we will extensively research several methods for prediction and label uncertainty estimation in both Explicit and Implicit feedback-based Recommender Systems. The literature in the field is very disconnected, and for this reason, we chose to present the most relevant related work alongside our own. In every chapter, we will emphasize which of the presented methods are original to this dissertation. In addition, Tables 1.1, 1.2 and 1.3 list the methods we compare and highlight those that are novel. These chapters are organized as follows.

In Chapter 3, we explore and extend methods for prediction uncertainty estimation for explicit feedback-based RSs. This chapter includes a very extensive review and categorization of the research field. Furthermore, it also introduces two novel methods: ENSEMBLE and EB-Linear. Finally, it presents a comprehensive empirical comparison between the methods.

Chapter 4 is similar to the previous one, but it covers methods for prediction uncertainty estimation in Implicit feedback-based Recommender Systems. It generalizes many of the methods from Chapter 3 to the case of IRSs. In addition, it discuss the use of methods for prediction uncertainty quantification in neural network based IRSs, including two that have not been previously explored in the RS literature: Bayesian Neural Networks and Deep Ensembles. It ends with another empirical comparison.

Chapter 5 introduces a collection of novel methods for label uncertainty estimation in implicit feedback-based RSs. More precisely, it introduces four new methods, ICPMF, Gaussian-MLP, Gaussian Binary Ranking and Hierarchical Binary Ranking, some of which involve extensive mathematical modeling work. Again, the chapter culminates in an empirical comparison between the four novel models and another one from the literature.

Finally, Chapter 6 presents our conclusions and discusses possible directions for future research.

Chapter 2

Recommender Systems

Recommender Systems are a response to ever-expanding choice and to information overload. An RS uses data and algorithms to predict which candidate items may appeal to its users, often making these predictions in a personalised or contextualised way [RRS11]. Ideally, it exposes its users to items (such as products, services, news articles or even people) that the users may not have discovered for themselves, and it helps users to manage the choice of which items to consume. RSs can be found in many different application domains, often employed at scale: for example, we find them in music and movie streaming services, in online shopping sites, in social media platforms, and in news story aggregators.

In this chapter, we will discuss the data and algorithms that RSs use, and also the topic of RS evaluation. The field is huge and a comprehensive survey impossible. Our decisions over selection of material are guided by topics that underpin later chapters of this dissertation.

We exclude from this chapter papers that propose specific methods for estimating the prediction and label uncertainty of recommendations, and the evaluation of these uncertainty estimates. These papers are covered in detail in subsequent chapters, where we bring our own contributions by: expressing these methods in a unified notation; critiquing them; proposing new methods; and including them in comprehensive empirical comparisons.

2.1 Data in Recommender Systems

In practice, an RS relies on diverse types of data to offer personalized suggestions. The types of data available guide the choice of recommendation algorithms. We

split data into two: user-item interaction data, and non-interaction data. This somewhat high-level division reflects the fact that the emphasis in this dissertation is on interaction data. Accordingly, we review non-interaction, which we make no use of in the rest of this dissertation, in a more cursory way. We further subdivide the user-item interaction data into two, as per the previous chapter: explicit feedback and implicit feedback.

2.1.1 User-item interaction data

User-item interaction data is a record of a user's response to an item, e.g. one that she was exposed to or one that she consumed. Usually, it takes the form of explicit or implicit user feedback (see below). However, there could be other types of user-item interaction data; for example, a timestamp or contextual data such as the weather or location of the user at the point when the interaction was observed.

User-item interactions are the building blocks of Collaborative Filtering (CF). CF is one of the most notable classes of recommendation algorithms. In short, these algorithms recommend based on the preferences and behaviours of other similar users. They rely on finding patterns in user-item interactions to make personalized recommendations. In Section 2.3, we will explore CF in more detail.

2.1.1.1 Explicit user feedback

Often, after a user has consumed an item, an Explicit feedback-based Recommender System (ERS) may explicitly invite the user to *rate* the item. Traditionally, *explicit feedback* is given in the form of a star rating, i.e. a value chosen from a discrete scale, e.g., 1⋆ to 5⋆. The higher the rating, the more the user liked the item. For our work on explicit feedback in this dissertation, we use star ratings, but we note that there are other possibilities: explicit feedback could be binary (e.g. thumbs-up or thumbs-down) or even unary (where only thumbs-up is allowed).

The use of numeric ratings, and of explicit feedback in general, has several advantages. First and foremost, they quantify the user sentiment towards the item. For example, if a user rates movie A with 5⋆ and movie B with 4⋆, the system can easily understand that the user prefers A over B. Furthermore, ratings can designate negative sentiment (e.g. where 1⋆ means that the user did not like the item). Finally, the fact that they are explicit also makes them more transparent: the users can choose which items they want to rate.

On the other hand, there are also several widely-recognized issues involving explicit ratings. First, users are not always consistent with the way they rate items. In [APO09], it was noted that, when prompted to re-rate previously-rated items, users often provided a value that differed from their original rating. Secondly, the user's choice of *what* to rate and *how* highly to rate it might be biased by a variety of social factors [CLA+03]. For instance, a user might provide higher ratings to critically-acclaimed items.

In addition, explicit feedback is not easy to gather. The users must be willing to re-engage with the system, subsequent to consumption, by actively providing ratings in the expectation of receiving better recommendations in the future. For this reason, ERSs are often bottle-necked by rating non-availability. Some ERSs seek to overcome this problem using Active Learning (AL) [Car20], in which the system queries users for their feedback on items they are known to have consumed or predicted to have consumed, where the users and items are selected in the expectation of having an impact on system performance.

2.1.1.2 Implicit user feedback

There are other forms of feedback that are not explicitly given by the user. Some examples are searches, clicks, views, purchases, favourites, etc. These are referred to as *implicit feedback*. Usually, implicit feedback is unary: all the feedback received is perceived as positive and equal, and this is what we will assume in this dissertation. However, in some cases, there may be degrees of importance assigned to implicit signals [LJ14]. For instance, in online shopping, a click action is less important than an add-to-cart action, which, in turn, is less important than a purchase action.

The advantages of implicit feedback are plenty. For one, it is much easier to collect: it is recorded as a side-effect of normal user interaction, rather than through a separate dialog. In addition, implicit feedback does not suffer from the aforementioned inconsistencies involving interpretation of rating scales.

On the downside, since it is most often unary, implicit feedback does not allow for negative sentiment. Where a user has been exposed to an item but did not interact with it, there is no clarity on whether the user found the item to be unappealing or whether she did not interact with the item for some other reason [HKV08]. Also, even when there is an interaction it may not signal positive feedback, e.g. a mistaken click.

Historically, the RS literature focused on ERS. This may be because the field was strongly driven by the Netflix Prize contest [BL07] and the MovieLens system[1], both of which provided the research community with datasets of numeric ratings. Subsequently, the enormous availability of implicit feedback in deployed systems, combined with its aforementioned advantages, has led to a drift in the field towards Implicit feedback-based Recommender Systems (IRSs). Furthermore, it has been shown in several cases that recommendations from models trained on implicit feedback are more satisfactory to the user [ZHAK18].

2.1.2 Non-user-item interaction data

There are many other types of data. Since they are not the focus of this dissertation, we only summarize them below.

User demographic data: User information, such as age, gender, location, and other relevant attributes, can offer valuable insights for personalizing recommendations. By incorporating this data into recommendation algorithms, systems aim to provide more tailored and relevant content to users. In many cases, the use of user demographic data, along with user-item interactions, has proven effective, leading to recommendations that are more personalized and more accurate [Kor08, GPB10, GTY+17]. In particular, demographic data can be very helpful for cold-start users, who are novel to the system [SS13].

Item data and meta-data: An RS may also use data that describes the items in its catalog. In the case of digital media, such as music, images, movies or e-books, this data might be extracted or mined from the digital format itself. More often, it takes the form of meta-data: descriptive attributes or features associated with the items, including text descriptions, genres, categories, keywords, user-assigned tags[2], attribute-value pairs and so on. Item descriptions, often referred to as item content, allow for a special class of recommenders to be used, known as Content-Based (CB) recommenders [LDGS11]. In overview, CB RSs compare the descriptions of candidate items with the contents of user profiles, which record descriptions of the items that the user likes. CB RSs can perform well when there is abundant item information available. An advantage is their ability to work well even

[1] `https://movielens.org/`

[2] If the tags for an item are aggregated, then they are an example of item meta-data; but if we preserve user-item-tag relations, then arguably this is an example of implicit feedback data.

for cold-start users and infrequent users, who have provided no, or little, feedback data [LKH14]. However, these systems face challenges when it comes to item discovery in parts of the catalog that are fresh to the user because they tend to recommend items that are closely related to those previously consumed.

Knowledge graphs: Knowledge graphs offer a promising avenue for improving recommendation quality by incorporating structured information about users, items, and their relationships. By enriching the representation of users and items with complex domain-specific knowledge, such as attributes and contextual information, the RS might deliver more context-aware and semantically meaningful recommendations. For an extensive review on the matter, we refer to [GZQ+20].

Social networks: Social networks are a special case of knowledge graph that may be incorporated into a RS. Users often trust the opinions of their contacts more and are more similar to them due to social influences. An example of the usefulness of social networks to recommendation is presented in [MZLK11].

Context data: Context data is any additional data that influences a user's preferences in a specific situation [AT10, VMO+12]. Examples include the device the user is using, the consumption time and location, whether the user consumed the item alone or with companions, and so on.[3] By incorporating context data, an RS can match recommendations to the user's current context to provide suggestions that are contextualised, as well as personalised. Systems that utilize context data are known as Context-Aware (CA) RSs.

2.2 Recommendation Workflow

The ultimate task of an RS is to provide each user with a top-n set of *personalized* and, perhaps, *contextualized* recommendations. Conceptually, we can view this as a three-step process: candidate selection, scoring, and top-n recommendation. This is illustrated in Figure 2.1. Of course, this is not intended as an RS architecture: real systems will be much more complex. Rather, it offers a way of structuring the presentation of the recommender workflow.

Candidate selection: The first step in the workflow is to select which subset

[3]There is an argument that context data is another form of user-interaction data, since it is often related to the consumption of an item by a user.

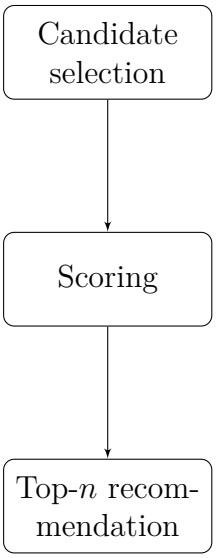


Figure 2.1: Recommendation workflow.

of items from the catalog are *plausible* recommendations. Items may be implausible for many reasons. Take as example a restaurant recommender for use on-the-move. Naturally, only restaurants in the proximity of the user should be considered. Another example is news recommendation. In this case, most news stories that are more than, let's say, a week old are irrelevant, and therefore should be removed from the candidate pool. Candidate selection is also key in achieving good response times. When the item catalog is huge, scoring all the available items might be too time consuming. One solution is to filter a subset of candidate items. Such filtering must be performed carefully, to avoid discarding too many items that could be relevant to the user.

Scoring: With a filtered set of candidate items at hand, the system must now rank these items according to some criteria. A scoring function is defined as $F : U \times I \rightarrow \mathbb{R}$. For each user and item, it can give a real-valued score, such that if $F(u, i) > F(u, j)$, user u is predicted to prefer item i over item j. Scoring is unarguably the most researched matter in the RS field. When explicit ratings are available, scoring is done through *rating prediction*. When they are not, we resort to *relevance prediction*. In Section 2.3, we extensively describe methods for scoring in both explicit and implicit feedback-based recommenders.

Top-n recommendation: The final step is to select an n-sized subset of candidate items to present to the user. Often this is done by simply sorting the candidate items in decreasing order of their scores, and then selecting the

first n from the sorted set. However, this is not the only way of choosing the n items. In some cases, for example, the system may choose one or more items that score lower in the sort order. This may be done to satisfy so-called beyond-accuracy objectives, such as diversity and serendipity. In Section 2.4, we explain several common beyond-accuracy objectives and how top-n recommendation can be performed in a way that favours them.

We will not say any more about candidate selection since it is usually domain-specific, often quite simple relatively speaking, and of lower relevance to the remainder of this dissertation. The next two sections present scoring and top-n recommendation in more detail.

2.3 Scoring

Scoring plays a pivotal role in the RS workflow, serving as the critical bridge between the pool of candidate items and the final set of personalized recommendations. In essence, scoring involves the data-driven assessment of each candidate item's relevance to each user's inferred preferences. The scoring algorithms, which aim to assign numerical values to candidate items, are plentiful. They can be divided into two main groups: Content-Based (CB) and Collaborative Filtering (CF). As previously stated, CB RSs rely on item description data, and will not be further explored in this dissertation.

CF leverages the wisdom of the crowd. It bases recommendations on the interaction data of other users and items. CF algorithms can be further divided into two sub-types: memory-based and model-based. Memory-based approaches, such as user-based and item-based collaborative filtering, use nearest-neighbour techniques to predict item ratings or item relevance from the user-interaction data of similar users or items. Model-based approaches, such as matrix factorization and neural network methods, use machine learning techniques to learn latent factors that represent user-item interactions.

CF is advantageous because it can infer subtle user preferences, especially when large amounts of feedback are available, either explicitly or implicitly given. However, CF suffers from the *sparsity* problem. Its performance may not be so good when there is little user-interaction feedback available or when users' preferences diverge significantly.

Hybrid systems can be created by integrating different recommendation methods, e.g. collaborative filtering and content-based [Bur02b]. Given that different

recommender scoring algorithms work well in different scenarios, hybrid systems are a way of combining their strengths.

In this dissertation, as we have already said, the only data that we work with is user feedback data. Naturally, then, we will focus on CF methods and, more specifically, on model-based CF. There is a vast number of specific models, including methods to work with explicit feedback and methods to work with implicit feedback. The next few subsections present some popular CF methods — ones that will be used in later chapters of this dissertation.

2.3.1 Modeling user-item interactions

The power of an RS comes from its ability to understand the users and the items in the system based on the available interactions, as well as any other data such as item descriptions. In several areas of machine learning, it is useful to learn latent representations for entities such that the system can understand how they relate to each other. RSs are no exception to this. Many model-based CF recommendation algorithms learn latent representations for their users and items.

One of the first RSs based on latent representations, and the most influential, is Matrix Factorization (MF) [Fun06]. The idea is to represent users and items in a low dimension latent space. If the number of latent factors is denoted by d, then each user u is represented by a vector p_u and each item i by a vector q_i, both of dimension d. Then, in the case of explicit feedback, we can compute predicted ratings by taking a dot product:

$$F_\theta(u, i) = p_u^T q_i \tag{2.1}$$

where $\theta = \{\{p_u\}_{u \in U}, \{q_i\}_{i \in I}\}$ is the parameter set of the model. This seemingly simple model has proven effective, and became famous, by emerging during the Netflix recommendation challenge [BL07]. The interactions in the Netflix challenge were numeric ratings. But the same model can also be used in IRS by scaling the output properly [HKV08]; see Section 2.3.3.

Recent research has shown that the inner product in Eq. 2.1 is a very accurate way of representing user-item interactions and is not easily outperformed even by recent neural neural network models [RKZA20]. Nevertheless, there are many extensions to the simple dot-product model. The simplest one is adding user and

item biases to the equation. That is,

$$F_\theta(u,i) = p_u^T q_i + b_u + b_i \tag{2.2}$$

where b_u and $b_i \in R$ are scalars that model each user's and item's individual contributions to the prediction, and $\theta = \{\{p_u, b_u\}_{u\in U}, \{q_i, b_i\}_{i\in I}\}$. Other notable extensions include SVD++ and AsymmetricSVD [KBV09].

More recently, Neural Network (NN) based models have gained traction in the RS field. Several such models have been proposed [CKH+16, HLZ+17, NMS+19], with some achieving impressive results. Several properties of neural networks explain their increasing use in this field. First, the flexibility of neural architectures allows for a wide variety of data inputs, making it relatively straightforward to combine interaction data with, e.g., user demographic data, item descriptions, and contextual data [GTY+17]. Second, neural networks are a great tool for latent representation learning, as shown by the success, for example, of variational auto-encoder recommenders [LKHJ18].

Later in this work, we will utilize one of the first and most popular NN based recommenders, the Multi-Layer Perceptron (MLP), introduced in [HLZ+17]. Similarly to MF, this model also learns d-dimensional user and item embeddings, p_u and q_i, but it then combines them through a Multi-Layer Perceptron neural network,

$$F_\theta(u,i) = \text{MLP}(p_u || q_i) \tag{2.3}$$

where $||$ is a concatenation operator and $\theta = \{\{p_u\}_{u\in U}, \{q_i\}_{i\in I}, \theta_{MLP}\}$, θ_{MLP} being the parameters of the MLP. The MLP consists of a set of feed-forward layers $f_1, \ldots, f_L$, such that,

$$f_0 = p_u || q_i \tag{2.4}$$

$$f_l = \text{ReLU}(W_l f_{l-1}), \quad \text{for } l \in 1, \ldots, L-1; \tag{2.5}$$

$$f_L = g(w_L^t f_{L-1}) \tag{2.6}$$

where W_l is the weight matrix for hidden layer l, w_L is the output layer's weight vector, and $\text{ReLU}(x) = \max(0, x)$. In the output layer f_L, an activation function g is used to scale the output to the desired interval (often to [0, 1] in the case of IRS).

2.3.2 Learning from explicit feedback

In ERS, the known ratings are indicative of the users' preferences. Therefore, scoring can be performed through *rating prediction*. Given a user $u \in U$ and item $i \in I$, the rating prediction task consists of designing or learning a function $F : U \times I \rightarrow \mathbb{R}$, parameterized by θ that predicts user-item ratings.

One way to learn the model parameters θ (the user and item embeddings, p_u and q_i and any other parameters) is through gradient descent: their values are initialised with randomly-chosen values; the algorithm proceeds by sampling instances from the training set of known ratings, calculating the difference between the predicted and the actual ratings for these instances; and updating p_u and q_i through gradient descent to minimize these errors. In ERS, the most commonly used loss function is the Mean Squared Error (MSE). In this case, given a batch of training instances $\mathcal{B}$, the loss to be minimized is:

$$\mathcal{L}(\mathcal{B}|\theta) = \frac{1}{\#\mathcal{B}} \sum_{(u,i,r_{ui})\in\mathcal{B}} (r_{ui} - F_\theta(u,i))^2 + \lambda||\theta|| \tag{2.7}$$

where λ is a regularization hyper-parameter, used to avoid overfitting, $||\cdot||$ denotes the Frobenius norm, and $\#$ denotes the cardinality of the set.

A Matrix Factorization model learned with the MSE loss was popularized as FunkSVD [Fun06], after its success at the Netflix prize challenge [BL07].

2.3.3 Learning from implicit feedback

Implicit feedback differs drastically from the explicit case. In the latter, the relevance to the user of a given candidate item can be given by the prediction of a numeric rating. But, for the former, all the observed interactions are assumed to be equally relevant, even if this assumption is a simplification. Furthermore, with implicit unary data, we have no negative sentiment to learn from. For the purposes of learning, it is common to assume that unobserved interactions are less relevant than observed ones. This is a key assumption that underpins the learning objectives that will be explored in this section. Naturally, this assumption is also a simplification because unobserved interactions are very often only a case of the user having never been exposed to the item; other times, they are cases where the user was exposed to the item but, for whatever reason, did not interact with it. In summary, from the available implicit data, we take it that all interactions in R, that is observed interactions, are of equal relevance among themselves and

of higher relevance than those in $R^- = \{(u,i)|u \in U, i \in I, (u,i) \notin R\}$, the unobserved interactions.

In the absence of explicit ratings, we must score the candidate items by inferring their *relevance* to the user. Given a user $u \in U$ and item $i \in I$, the *relevance prediction* task consists in designing or learning a function $F : U \times I \rightarrow \mathbb{R}$, parameterized by θ, that predicts user-item interaction relevance. The predominant objectives that suit implicit data can be categorized in two classes: point-wise and pair-wise. Below, we explore each of these classes. Nevertheless, we acknowledge that there are other classes of objectives, especially list-wise ones [SLH10], that will not be explored in this dissertation because they are less popular.

2.3.3.1 Point-wise objectives

Point-wise objectives, as the name suggests, score data points (interactions) individually. That is, during learning, the model's current prediction for the interaction is compared against an observed value. Now, because implicit feedback consists of unary observation data, we define the following labels for the interactions,

$$Y_{ui} = \begin{cases} 1, (u,i) \in R \\ 0, (u,i) \in R^- \end{cases} \tag{2.8}$$

Hence, learning from implicit feedback data has one major particularity. In the explicit case, the training set consists only of observed ratings (denoted R). In other words, unobserved interactions are treated as *missing*. This is not the case for implicit data, where the unobserved interactions are part of the training set (R^-, labeled as 0's), along with the observed interactions (R, labeled as 1's).

Using this labeling (Eq. 2.8), the implicit feedback task closely resembles a classification task, with observed interactions treated as ones, and non-observed ones treated as zeros. Therefore, the same objective functions used in classification tasks can be used [HKV08].

A very natural choice is to use the same MSE loss from before, but now on the labels from Eq. 2.8. An advantage is that it allows a model to be learned based from implicit data in the same way as with explicit ratings. In this case, the loss function be to be minimized is,

$$\mathcal{L}(\mathcal{B}|\theta) = \frac{1}{\#\mathcal{B}} \sum_{(u,i) \in \mathcal{B}} (Y_{ui} - F_\theta(u,i))^2 + \lambda ||\theta|| \tag{2.9}$$

where, again, λ is a regularization hyper-parameter that controls overfitting on the model's parameters θ. While this is an easy approach, this objective treats the labels given to the observed and unobserved interactions as numerical values of 0 and 1, rather than as class labels.

A more sound approach is to predict the probability of observing each label. Binary Cross-Entropy (BCE) gives one way of doing this.

Formally, let $P(Y_{ui} = 1) = g(F_\theta(u, i))$, where g is a function that bounds the model's outputs to $[0, 1]$, so that the outputs are well defined probabilities. Here, we will use the Sigmoid function, defined as $g(x) = 1/(1 + e^{-x})$, but there are many other possible choices. Then, the likelihood of the observed data is,

$$\mathbb{P}(R|\theta) = \prod_{u \in U} \prod_{i \in I} \mathbb{P}(Y_{ui} = 1)^{Y_{ui}} \mathbb{P}(Y_{ui} = 0)^{1 - Y_{ui}} \tag{2.10}$$

where $\mathbb{P}(Y_{ui} = 0) = 1 - \mathbb{P}(Y_{ui} = 1)$. For numerical stability, during learning, we minimize the negative log-likelihood, subject to L2 regularization to avoid overfitting.[4] In this case, the loss function for a data batch $\mathcal{B}$ is,

$$\mathcal{L}(\mathcal{B}|\theta) = \frac{1}{\#\mathcal{B}} \sum_{(u,i) \in \mathcal{B}} -Y_{ui} \log \mathbb{P}(Y_{ui} = 1) - (1 - Y_{ui}) \log \mathbb{P}(Y_{ui} = 0) + \lambda ||\theta|| \tag{2.11}$$

In practice, regardless of the specific objective chosen, the parameters are learned by mini-batch gradient descent, minimizing the average batch loss on each step. One problem, though, is that the data is usually very imbalanced. This happens because the set of unobserved interactions R^- is, in general, much larger than the set of observed ones R. To alleviate this issue, a sampling method is used. On each training epoch, the training data consists of the observed interactions R and an $(\#R \times N)$-sized randomly-selected sample of non-observed interactions from R^-, where N is a hyper-parameter. Algorithm 1 formally describes the method of sampling used with point-wise learning objectives.

2.3.3.2 Pair-wise objectives

Pair-wise objectives score pairs of interactions to learn the most plausible ranking. Broadly, they are based on the principle that observed interactions should always have higher relevance than unobserved ones. Here, we will cover only the most popular among these objectives, Bayesian Personalized Ranking (BPR), but there

[4] This form of regularization is equivalent to the input of spherical priors under the model parameters [MS08].

Algorithm 1: Learning a CF model from implicit feedback with Eq. 2.9 or 2.11.

Data: The observed and unobserved set of interactions: R and R^-.
Input: N the number of negative samples per positive; λ: the regularization strength; η: the learning rate.
Result: F: The learned scoring function.

```
Initialize model parameters θ;
while not converged do
    Sample an (#R × N)-sized batch of negative samples, B⁻, from R⁻;
    Construct the training data for the current epoch as B = {R, B⁻};
    for each mini-batch b ⊂ B do
        Compute the loss for the batch L(b|θ);
        Compute the gradient with respect to the loss;
        Update model parameters using gradient descent with learning rate η;
```

are others. One notable example is LambdaRank [Bur10]. For comprehensive reviews on the matter, refer to [BRL06, BC12].

Let $F_\theta(u, i)$ denote the relevance score of item i to user u. For ease of notation, let $F_{uij} = F_\theta(u, i) - F_\theta(u, j)$. Then, F_{uij} models how much user u prefers item i over j. In BPR, given a user u and a pair of items (i, j), the probability that the user u will prefer item i over j is modeled as.

$$\mathbb{P}(i >_u j) = \text{Sigmoid}(F_{uij}) \tag{2.12}$$

Again, the Sigmoid function is used to constrain model outputs to the $[0, 1]$ probability interval. Then, the likelihood of all the observed preferences is given by,

$$\mathbb{P}(R) = \prod_{u \in U} \prod_{i \in R_{u\cdot}} \prod_{j \notin R_{u\cdot}} \mathbb{P}(i >_u j) \tag{2.13}$$

The BPR loss function is the negative log-likelihood, subject to L2 penalty. It has been shown that optimizing the full likelihood is very slow [RFGST09]. Instead, a better approach is to sample positive-negative interaction pairs randomly and perform stochastic gradient updates. Therefore, the loss for each sampled interaction pair is,

$$\mathcal{L}(u, i, j|\theta) = -\log(\mathbb{P}(i >_u j)) + \lambda||\theta|| \tag{2.14}$$

Algorithm 2 describes the BPR learning procedure.

Algorithm 2: Learning a CF model from implicit feedback with Bayesian Personalized Ranking (Eq. 2.14)

Data: The observed and unobserved set of interactions: R and R^-.
Input: λ: the regularization strength; η: the learning rate.
Result: F: The learned scoring function.

```
Initialize model parameters θ;
while not converged do
    Sample one interaction (u, i) from R;
    Sample one negative interaction (u, j), for the same user, from R⁻;
    Compute P(i >_u j) with Eq. 2.12;
    Compute the negative log likelihood according to Eq. 2.14;
    Update model parameters using gradient descent with learning rate η;
```

2.4 Top-n Recommendation

The last step of the recommendation workflow is to select a top-n set of items to be presented to the user: for a user $u \in U$, the top-n recommendation task consists of selecting $Z_u^n \subseteq C_u$, which is the set of n items from C_u that are predicted to be most appealing to u.

The scoring functions from the previous section, either in the form of predicted ratings (explicit feedback) or predicted relevance scores (implicit feedback), allow the system to rank items: the higher the predicted value for an interaction, the more pertinent we expect the item to be for the user. The top-n recommendation task is conventionally performed by sorting the candidate items C_u in descending order of their predicted ratings or relevance scores and then forming Z_u^n by selecting the top-n items from this sorted set. We will refer to these ways of doing top-n recommendation as Rating-Based Ranking (RaBR), in the case of ERS, and Relevance-Based Ranking (ReBR), for IRS.

Nevertheless, in many cases, and for many reasons, a system might use alternative methods for obtaining the top-n. For example, it may be that some 'slots' in the top-n must be reserved for sponsored items. Or, in a hybrid RS, it may be necessary to combine the scores of the models that are part of the hybrid before sorting and selecting the top-n, or even to ensure a mix of items that get high scores from the models that are part of the hybrid [Bur02a]. In addition, for systems where *beyond-accuracy* recommendation objectives are important, many researchers have proposed methods for *re-ranking*[5] [KB16]. In summary, these

[5]Re-ranking is a tool to incorporate beyond-accuracy objectives into the top-n recommendation stage of a RS. Nevertheless there are ways of incorporate these objectives into other stages, e.g. the scoring stage [BBC20, DMRAB+19]

methods aim to modify the order, or even change items in the recommendation list, in order to increase some beyond-accuracy criteria.

[KB16] presents a comprehensive review of the most-researched beyond-accuracy recommendation objectives. In the upcoming sections, we will introduce the concepts of diversity, serendipity, novelty and coverage. In addition, we will also discuss fairness and explainability. These two are not listed in [KB16], but one can argue that they do deserve to be treated as additional beyond-accuracy objectives.

Later in this dissertation, we will also introduce novel forms of ranking that are uncertainty-aware: The final set of top-n recommendations can then take into account the estimated prediction or label uncertainties of the recommendations.

2.4.1 Diversity

Top-n diversity in recommender systems refers to the extent to which a recommendation list contains a wide range of distinct items, aiming to offer users a variety of choices. It is a measure of how dissimilar or different the items in the top-n recommendations are from one another, and it plays a crucial role in enhancing user satisfaction and exploration of new content. In essence, diversity ensures that users are presented with a balanced and heterogeneous set of recommendations rather than a narrow selection of similar items.

Many items in a system's catalog tend to be similar. For example, in a music RS, many songs are from the same artist; in a movie RS, many movies will share genres. Therefore, the system may provide a list of recommendations composed of items that are similar among themselves, that is, a non-diverse recommendation list. The recommendations might be accurate, for example, recommending several songs from the same artist or movies of the same genre if the user is known to like that artist or genre. But, such low diversity means that the RS does not (i) expose the user to much of the vast product catalog, and (ii) offer the user a *meaningful choice* of recommendations.

We remark that, in this section, we present the most common definition of diversity in RSs, but there are others. One example is *temporal diversity*, which refers to the difference in two recommendation lists given to the same user at different time points. For a more extensive review, see [KB16].

2.4.2 Serendipity

Serendipity is a concept that derives from *surprise*. In RS, surprising recommendations are those that deviates from the user's expectations. Giving surprising recommendations is not necessarily hard. For instance, based on some item distance metric, items that are the furthest from the items in the user profile would likely be very surprising to the user. Nevertheless, these same surprising items would most likely not be relevant. For this reason, serendipity goes one step further from surprise. A serendipitous recommendation can be thought as a surprising, yet relevant, item.

Giving relevant recommendations is already a hard task in most cases. Now, finding relevant and surprising recommendations is a lot harder. But the difficulty of the process should not stop the system from trying to find them. Serendipity brings unexpected and delightful discoveries to users. Systems that prioritize serendipity can help alleviate recommendation fatigue, a situation where users are repeatedly shown the same type of content. By introducing serendipitous items, users are less likely to grow tired of the recommendations and are more likely to return to the platform [FX22]. Ultimately, serendipity contributes to overall user satisfaction. Users appreciate platforms that not only cater to their known preferences but also offer exciting, unanticipated recommendations that add value to their experience [CYW+19].

2.4.3 Novelty

The concept of novelty is closely related to serendipity. In fact, in the recommender systems literature, the terminologies often overlap. According to many authors, novelty also refers to recommendations that are not closely related to the user profile [VC11, Zha13, CHV21]. Nevertheless, other researchers distinguish the concepts. According to [HKTR04], a *novel* recommendation is not necessarily *surprising*. Take as example a movie recommender, where a certain user u frequently consumes action movies. Now imagine u gets recommended one action movie which they are unaware of. Such a recommendation is novel, but not surprising.

In summary, novelty and serendipity share the idea that the item is *unknown* to the user, but do not share the element of *surprise*. In addition, [KKT+15] extends the notion of novelty in the case of systems that support repeat consumption, e.g. music recommenders. In these sorts of systems, a novel item is not necessarily unknown by the user, but could be forgotten.

2.4.4 Coverage

User coverage refers to the fraction of the users in the system to which the recommender is able to provide recommendations. Under this definition, coverage differs from the previous concepts in this section: it does not refer to individual recommendations or sets of recommendations, but instead, it reflects the recommender's ability to reach its users. There are several reasons why users might not be covered by the system. For example, as previously stated, CF-based RSs depend on interaction data, and are, therefore, unable to provide recommendations to extreme cold-start users – that is, those who have not interacted with any item. Another possible cause for loss of coverage is computation delay: when the system is unable to compute recommendations in time [LRS+17].

Symmetrically, item coverage is the fraction of the item catalog that can get recommended. Item coverage is very dependent on the choice of recommendation algorithm. One extreme example is popularity-based recommenders: these systems simply recommend the most popular items to every user. In this case, only a tiny fraction of the items are ever recommended. Several other methods also suffer from popularity bias: the more popular an item is, the more it gets recommended. In fact, the mitigation of popularity bias is an active research field in RS [ABM19, AABA22].

Finally, there is a third notion of coverage, which can, in fact, be defined for recommendation lists. In this case, it is the fraction of the desired number of recommendations that the system is able to provide to a user. Under this definition, the notion of coverage considers the possibility that users may be only partially covered. That is, if a system is programmed to present the top-n recommendations to user, there may be cases where the system may only recommend $m < n$. This particular definition is relevant to Uncertainty-Based Filtering (UBF), a ranking method that we will explore throughout this dissertation.

2.4.5 Fairness

Fairness in recommender systems encompasses multiple dimensions, depending on whether we view it from the user or item perspective. Recognizing these distinctions is essential to address biases and ensure equitable recommendations.

From the user perspective, fairness refers to treating users impartially, regardless of their demographic characteristics or preferences. Fairness entails avoiding algorithmic biases that could disproportionately impact certain groups.

From the item perspective, fairness involves providing visibility to all items in the system. A fair system should, for example, avoid the popularity bias – the case where popular items are recommended very frequently and unpopular ones get almost no exposure. In addition, the system must avoid disfavoring certain item providers. A concrete example of unfairness happens in the music domain, where female singers get recommended less than male singers [MRPC+21].

There literature investigating fairness in RS is growing. For a comprehensive survey on the matter, we refer to [WMZ+23]. Recently, [WJ23] proposed the utilization of uncertainty estimates to ensure fairness in recommenders.

2.4.6 Explainability

Explainability has been a hot topic of discussion in all fields of machine learning in recent years, and the RS field is no exception. Explainability in recommender systems can refer to the capability of providing transparency or interpretable reasons for why a specific recommendation was made. It involves making the recommendations, and perhaps the recommendation process, understandable to users, allowing them to comprehend why a particular item was suggested to them. Due to the increasing importance given to explainability, the literature in the field is evolving rapidly. Here, we only briefly describe its importance. A comprehensive survey on methods for explainability in RS can be found in [ZC+20].

Accompanying recommendations with an explanation has several purposes [TM10]. From the user point of view, explanations that increase transparency might also increase the user's trust in the system. Explanations might also increases persuasiveness: a well put explanation may persuade users to believe that certain items are relevant to them [BSDN+18].

From the system point of view, explainability can assist in identifying and addressing biases in recommendation algorithms. By revealing the factors that influenced a recommendation, it becomes easier to detect and rectify any undesirable biases, ensuring fairness and ethical use of recommendation systems [AN18].

2.5 Evaluation

In the evaluation phase of the development of a new RS, the quality of the system's predictions and recommendations must be rigorously assessed. There are three

primary evaluation approaches, *offline*, *online* and *user trials*.

The offline evaluation of an RS utilizes historical data. Much like other traditional ML tasks, such as regression or classification, to evaluate a RS offline, a *test set* of known ratings or interactions is initially hidden while learning the scoring model from the remaining data. Then, this test set is used to assess the quality of given predictions and recommendations. The hold-out test set may be selected in several ways, we list the three most common:

Random sampling: The case where every interaction from R is sampled with equal probability.

User-based sampling: A certain amount of interactions m from each user is sampled to compose the test set. For each user the m interactions are sampled at random from the user profile $R_{u\cdot}$. To account for varying profile sizes, another variation of user-based sampling instead samples a certain fraction of the interactions in each user profile.

Chronological sampling: In practice, user interactions are influenced by their past experiences. Therefore, in order to reflect reality, chronological sampling consists in taking the m latest interactions for each user as the test set. Similarly to user-based sampling, a fraction of each user's profile can be taken instead.

Offline evaluation is typically the first form of evaluation that a RS is subject to. This is mainly because it is cost-effective and efficient since it does not require real-time user interactions. In addition, it allows for the testing of various algorithms and configurations in a controlled environment.

Online evaluation takes the evaluation process beyond the controlled environment of offline testing and ventures into the real-world interactions between users and the recommender system. It consists in deploying the recommendation system in a live environment and continuously collecting data on user interactions with the system. It better reflects the dynamic nature of a RS by assessing the algorithm's performance in a real-world setting. Another advantage is enabling adaptive algorithms that can continuously learn and improve. Despite its benefits, evaluating a system online is not straightforward. Conducting live experiments can affect user experiences, potentially leading to user dissatisfaction if recommendations are not optimal. In addition, online evaluation can be resource-intensive and potentially expensive. For these reasons, academic research often resorts exclusively to offline evaluation.

Finally, user trials involve gathering feedback from trial participants who interact with the system in a controlled way. This typically includes surveys, interviews, or feedback mechanisms within the application for users to provide their opinions on the recommendations. By directly querying the users about their experience, user trials allow the system to gather qualitative feedback on user satisfaction and perception of recommendations. Unfortunately, user trials are even harder to implement, because they rely on user cooperation and willingness to provide feedback. In addition, user opinions may be more subjective, and therefore, harder to evaluate quantitatively.

Since this dissertation exclusively uses offline evaluations, the remainder of this section covers offline evaluation in a little more detail.

2.5.1 Evaluating rating predictions

In Explicit feedback-based Recommender System, scoring is performed through rating prediction. Therefore, predicted ratings can be compared against a held-out test set of known ratings. In this case, there are plenty of evaluation metrics to be borrowed from the regression literature. Among them, the most commonly used for rating prediction evaluation is the Root Mean Square Error (RMSE), mostly because it was the metric of choice in the Netflix prize challenge [BL07]. This metric simply compares the predicted ratings to the known ratings in the test set T, that is,

$$\text{RMSE} = \sqrt{\frac{\sum_{r_{ui} \in T} (F_{ui} - r_{ui})^2}{\#T}} \tag{2.15}$$

Other alternatives are the Mean Squared Error (MSE), which is the RMSE squared, and the Mean Absolute Error (MAE), defined as,

$$\text{MAE} = \frac{\sum_{r_{ui} \in T} |F_{ui} - r_{ui}|}{\#T} \tag{2.16}$$

2.5.2 Evaluating top-n accuracy

Of much greater importance than evaluating rating predictions is to evaluate the top-n recommendations [HKTR04, SG11]. Indeed, for IRS, where explicit numeric ratings are not available, top-n evaluation is the only option. Many metrics have been defined for evaluating the accuracy of top-n recommendations. In general, these metrics are shared with the field of information retrieval, due to its similarity to the top-n recommendation task.

First, consider a test set of (held-out) interactions T. Now, let T_u denote the set of items from users u's interactions in T. In the case of IRSs, $T_u = \{i|\langle u, i\rangle \in T\}$. But, for ERSs, $T_u = \{i|\langle u, i, r_{ui}\rangle \in T \cap r_{ui} \geq \omega\}$. In the latter, ω is a threshold, used to avoid rewarding the system for recommending an item in the test set that received a poor rating. For example, in a 5⋆ rating scale, ω might be 4⋆: if a system recommends a test item that the user had rated 4 or 5⋆, then the recommendation is regarded as having been relevant to that user.

Two traditional metrics are Precision@n and Recall@n. The precision and recall of the n-sized list of recommendations to user u, denote as Z_u^n, are given by:

$$\text{Precision@n}_u = \frac{\#\{Z_u^n \cap T_u\}}{n}$$

$$\text{Recall@n}_u = \frac{\#\{Z_u^n \cap T_u\}}{\#T_u}$$

The Precision@n and Recall@n of the system is reported as the average precision and recall among all users. That is,

$$\text{Precision@n} = \frac{1}{\#U} \sum_{u \in U} \text{Precision@n}_u \tag{2.17}$$

$$\text{Recall@n} = \frac{1}{\#U} \sum_{u \in U} \text{Recall@n}_u \tag{2.18}$$

Both precision and recall have a widely recognized issue when evaluating top-n recommendations: they give equal importance to all the recommended items. In practice, the earlier an item appears in a list of recommendations, the more attention a user is likely to afford it and, ideally, the more relevant it should be to the user. Therefore, evaluation metrics should account for position in the top-n. Here we define two of the most commonly used position-aware recommendation accuracy metrics.

Mean Average Precision: The MAP is a generalization of the traditional precision metric to account for the importance levels of the recommended items. To compute MAP, we first calculate the Average Precision (AP). AP is calculated by summing up the precision values at each point where a relevant item is found in the recommendation list and then dividing by the total number of relevant items. Formally, for each user u, the Average Precision

is calculated as,

$$\text{AP@n}_u = \frac{1}{\#T_u} \sum_{k=1}^{n} \text{Precision@k}_u \times \delta(Z_u^n(k) \in T_u) \tag{2.19}$$

where $Z_u^n(k)$ is the k-th recommended item and $\delta(Z_u^n(k) \in T_u)$ is an indicator function that equals 1 if the item at position k in the recommendation list is relevant for user u – that is, is in T_u –, and 0 otherwise. Finally, to obtain the Mean Average Precision across all users, we calculate the average of the AP values,

$$\text{MAP@n} = \frac{1}{\#U} \sum_{u \in U} \text{AP@n}_u \tag{2.20}$$

Normalized Discounted Cumulative Gain: NDCG measures the ranking quality of recommended items, accounting for the position of relevant items in the list. It gives higher scores to relevant items that are ranked higher. The metric is composed of two components:

- Discounted Cumulative Gain (DCG) calculates the cumulative relevance of recommended items, placing lower emphasis on items ranked lower in the list by using a logarithmic discount factor. DCG is computed as follows:

$$\text{DCG@n}_u = \sum_{k=1}^{n} \frac{2^{\delta(Z_u^n(k) \in T_u)} - 1}{\log_2(k+1)} \tag{2.21}$$

- Ideal Discounted Cumulative Gain (IDCG) represents the theoretical maximum possible DCG for a given set of relevant items. That is, if all items in T_u are recommended before any other item $j \notin T_u$.

Finally, the Normalized Discounted Cumulative Gain (NDCG) is computed by dividing the DCG by the IDCG, which scales the DCG value between 0 and 1, and the system NDCG@n is the average among all users:

$$\text{NDCG@n} = \frac{1}{\#U} \sum_{u \in U} \frac{\text{DCG@n}_u}{\text{IDCG@n}_u} \tag{2.22}$$

2.5.3 Beyond-accuracy top-n recommendation evaluation

As previously stated, there are several criteria, other than accuracy, that define what makes a good top-n recommendation list. Therefore, there is also a plethora of metrics to evaluate beyond-accuracy recommendation criteria. Here, we de-

scribe some traditional ways of evaluating diversity, serendipity, novelty and user coverage.

Top-n diversity: Top-n diversity can be measured in several ways. In general, all the measures will rely on some item distance metric. Let $d(i, j)$ denote the distance between items i and j. Then, the diversity in a list of recommendations can be calculated as the average distance between all the recommended items [KB16]. That is,

$$\text{Diversity}(Z_u^n) = \frac{\sum_{i \in Z_u^n} \sum_{j \in Z_u^n - i} d(i, j)}{n(n-1)} \tag{2.23}$$

Serendipity: Measuring serendipity is not straightforward, because it comprises surprise and relevance, two concepts that are hard to measure themselves. The general rule of thumb, as proposed by [GDBJ10], is to measure it as,

$$\text{Serendipity}(Z_u^n) = \frac{\#\{Z_u^{\text{surprising}} \cap T_u\}}{n} \tag{2.24}$$

In this equation, $Z_u^{surprising}$ denotes the subset of the recommended items that are surprising according to some defined metric. One possibility is to use the minimum distance to the items in the user profile, that is,

$$Z_u^{\text{surprising}} = \{i \in Z_u^n | \min_{j \in R_{u\cdot}} d(i, j) > \epsilon\} \tag{2.25}$$

where ϵ denotes the minimum distance for an item to be deemed surprising.

Novelty: Measuring novelty in recommenders is very challenging because it is practically impossible to infer whether a user is unaware of an item. One crude, non-personalised way to estimate novelty is to use the recommended items' popularity. The idea is, the less the item is interacted with, the more likely the item is to be novel [KB16]. Formally,

$$\text{Novelty}(Z_u^n) = \frac{\sum_{i \in Z_u^n} -\log_2 \left(\frac{\#R_{\cdot i}}{\#R}\right)}{n} \tag{2.26}$$

Coverage: Here we give a definition to the third notion of coverage that we mentioned in Section 2.4.4 — the one that applies to recommendations lists — since this is most relevant to later chapters of this dissertation. To measure the coverage of a recommendation list, one may simply compare the number of recommendations the system is able to deliver to the user against

the desired number of recommendations. Formally, let $\#Z_u$ denote the size of the recommendation list for user u and n the desired recommendation list size, then the coverage of the recommendation list is,

$$\text{Coverage}(Z_u) = \frac{\#Z_u}{n} \tag{2.27}$$

The average coverage is then obtained by averaging the individual coverage of each user.

2.5.4 Uncertainty in recommender systems

Uncertainty is present in every machine learning task. In particular, supervised learning tasks, such as scoring in RSs, are subject to prediction uncertainty [HW21]. We note that rating prediction is a regression problem. That is, a model is learned and used to predict a real-valued response variable: the rating for a given interaction. Similarly, and as previously stated, relevance prediction, particularly with point-wise objectives, resembles a classifier. As a consequence of the uncertainty around the predicted scores (ratings or relevances) that are used for ranking, the top-n recommendation stage of RSs is also subject to prediction uncertainty.

In addition to prediction uncertainty, IRSs also suffer from label uncertainty, which is introduced by the assumption of relevance (irrelevance) for observed (unobserved) interactions. In fact, a similar issue also appears in classification problems from the general ML literature [BF99].

Although this chapter is a literature review, we are not going to review papers that look at RS uncertainty here. Instead, we will present our review of existing relevant work alongside our novel proposed methods and in a consistent notation in subsequent chapters. However, the ML literature has many methods for uncertainty estimation, so we will review them briefly here and relate them to subsequent parts of this dissertation.

Probabilistic Regression: There are regression techniques that model the target variable in the form of probability distributions. From these distributions, it is possible to infer the uncertainty using dispersion metrics, such as their variance. One example is the CPMF recommender model, that we explore in Chapter 3.

Prediction Intervals: Instead of providing a single point prediction, regression

models can be used to generate prediction intervals, which express a range of values within which the true target variable is likely to fall with a certain level of confidence. Prediction intervals may be derived from probabilistic regression methods, or from distribution-free methods, such as conformal predictions [VGS05].

Residual Analysis: Analyzing the residuals (the differences between predicted and actual values) can help identify patterns of uncertainty in the model's predictions. EB-FunkSVD and EB-Linear (see Chapter 3) are examples of uncertainty estimation through residual analysis in RSs.

Ensemble Methods: Ensemble methods can provide an indication of uncertainty by combining the outputs of multiple regression or classification models [MMSJS12, RZS16]. We have ensemble-based recommenders in both Chapter 3 and Chapter 4.

Probabilistic Classification and Calibration: Some classification algorithms, such as logistic regression and naive Bayes, provide class probabilities. These probabilities represent the model's confidence in each class and can be used to assess uncertainty. Such probabilities are only useful if they are well calibrated [Daw82, P$^+$99].

We are now ready to turn to our own contributions to this field.

Chapter 3

Prediction Uncertainty in Explicit Feedback-Based Recommender Systems

Explicit feedback-based Recommender Systems have played a pivotal role in the evolution of recommender systems. Notwithstanding their usefulness to this date, these systems have historical importance due to their ability to directly capture users' explicit preferences and opinions. Unlike implicit feedback, where we infer preferences from user behaviour, explicit feedback involves concrete actions such as assigning a numerical rating or writing a review. For many reasons (see Chapter 2), implicit feedback has gained more attention recently. Nevertheless, there are several domains in which practical recommendation applications still rely, at least partially, on explicit feedback. Some examples include recommenders for restaurants, hotels, movies and TV shows, and many more. In most of the aforementioned services, explicit feedback is gathered in the form of numeric ratings, such as opinions on a 5-point scale. This is also the setting that we will confine our attention to in this chapter.

Due to the importance of ERSs to the field, both historical and current, this was also the setting adopted by [Maz13], which is one of the seminal works in prediction uncertainty[1] estimation for recommenders. Accordingly, we dedicate this chapter to reviewing and extending the methods for uncertainty estimation in ERS. More precisely, in this chapter, we:

[1]This whole chapter considers only prediction uncertainty, and not label uncertainty. For this reason, in this chapter, where we use the word 'uncertainty' on its own, it is prediction uncertainty that we are referring to.

- Present an extensive review of methods for estimating the uncertainty of rating predictions. These methods include information-based, stability-based, error-based, distribution-based and multinomial-based.
- Propose new techniques for estimating the uncertainty of rating predictions. In particular, we propose a new stability-based method, and also a new error-based method.
- Present an extensive review of the existing work on evaluating estimates of uncertainty for rating prediction and for top-n recommendation.
- Extend and improve the techniques and metrics for evaluating uncertainty estimates in RS. In particular, we analyse how uncertainty correlates with dataset statistics and we fix possible issues with existing metrics. We propose a new metric to evaluate uncertainty in the top-n recommendation task.
- Introduce the idea of uncertainty-aware ranking, drawing in part on a recommendation strategy proposed in [OLCGPB21]. In particular, we introduce and distinguish between Uncertainty-Based Filtering (UBF) and Probability-of-Relevance Ranking (PRR), and we give methods for evaluating these two top-n recommendation strategies.
- Conduct a reproducible empirical study on two large datasets and report results of the most extensive comparison of approaches to uncertainty estimation that we are aware of.

In the next section, we formalize the basic setting and notation that are specific for this chapter. In Section sec:terminology, we explain some of the roles that uncertainty estimates can play in rating prediction and top-n recommendation. In Section 3.2, we describe several methods for estimating uncertainty in ERSs. After that, Section 3.3 presents metrics for evaluating these uncertainty estimates. The remainder of the chapter evaluates the uncertainty estimates from Section 3.2 using the evaluation metrics from Section 3.3. In particular, Section 3.4 describes the experiments, while Section 3.5 shows the results. Section 3.6 discusses the uncertainty estimation methods in the light of the experiments. Section 3.7 concludes the chapter.

3.1 Prediction and Recommendation

As already mentioned, in this chapter, we deal with explicit feedback data – specifically, numerical ratings. Following from Table 1.4, we will designate the ordered set of possible ratings by $S = \{s_1, \ldots, s_m\}$, where m is the number of different possible rating values. For example, in the MovieLens dataset and the Netflix dataset that we use in Section 3.4, $S = \{1, 1.5, \ldots, 5\}$ and $S = \{1, 2, \ldots, 5\}$ respectively.

In addition, we will assume that our data consists of a set of users U, a set of items I, and a set R of user-item-rating triples, where triple $\langle u, i, r_{ui} \rangle \in R$ means that user u's rating of item i is $r_{ui} \in S$. A user's profile $R_{u\cdot}$ is the set of that user's ratings, $R_{u\cdot} = \{r_{ui} : \langle u, i, r_{ui} \rangle \in R\}$. Similarly, an item's support set $R_{\cdot i}$ is the set of that item's ratings, $R_{\cdot i} = \{r_{ui} : \langle u, i, r_{ui} \rangle \in R\}$.

3.1.1 Rating prediction

Given a user $u \in U$ and item $i \in I$, the *rating prediction* task consists in designing or learning a function $F : U \times I \to \mathbb{R}$ that predicts user-item ratings. That is, ranking prediction is the scoring stage of a recommender that is based on explicit feedback.

In this work, we want, not only to predict ratings, but also the *uncertainty level* present in a prediction. An uncertainty estimator is a function $G : U \times I \to \mathbb{R}$, that predicts real-valued quantities associated with each predicted rating, estimating how uncertain the system is about the rating prediction. We will denote by G_{ui}, the uncertainty estimate the system associates with predicted rating F_{ui}. In all the work that we report in this chapter, uncertainty levels are real-valued, $G_{ui} \in \mathbb{R}$. In some cases, they are probabilistic, which bounds them in $[0, 1]$, but this is not always the case. The higher the uncertainty level, the more uncertain the prediction.

Many of the methods for estimating uncertainty that we present in this chapter assume a rating prediction model F. Wherever this is the case, we will employ Matrix Factorization (MF) – see Eq. 2.1 – for this underlying model. This is known to be a fairly accurate predictor and, by using it in all cases where we need a separate predictor, we bring a degree of fairness to the comparisons.

3.1.2 Top-n recommendation

Of more importance than rating prediction is *top-n recommendation*. Given a user $u \in U$, the top-n recommendation task consists of retrieving Z_u^n from a set of candidate items $C_u \subseteq I$, such that the retrieved items are presumed relevant to the used.

In Section 2.4, we discussed the conventional way to perform top-n recommendation in ERSs, namely Rating-Based Ranking (RaBR), which consists of selecting the top-n based on the predicted ratings alone. But, given that we are considering models that can estimate prediction uncertainty, then there is an opportunity to consider other ways of selecting the top-n. These use what we can collectively refer to as *uncertainty-aware ranking*. If the estimates of uncertainty are good enough, we may achieve better (or, at least, usefully different) recommendation performance by recommendation strategies that rank the candidates using some combination of the predicted ratings and their estimated uncertainties. We propose two such strategies: *uncertainty-based filtering* and *probability-of-relevance ranking*.

3.1.2.1 Uncertainty-based filtering

In some domains, we may not want to recommend items if we are uncertain that the user will find them to be appealing. Whenever this is the case, uncertainty estimates may be used to prevent such recommendations. In the simplest case, we can use Uncertainty-Based Filtering (UBF). In UBF, we filter the candidate items, discarding those whose uncertainty is above some threshold τ.

If τ is small, it is likely that several of a user's candidate item pool will be filtered-out, which leads to more certain items being recommended. Some of these items may, however, be ones with smaller predicted ratings. Furthermore, some users might have a very small candidate set after filtering. For this reason, there might be cases in which the desired amount of recommendations cannot be provided to some users. On the other hand, larger τ values may mean that, for some users, few or even no items are filtered-out.

3.1.2.2 Probability-of-relevance ranking

If predictions are accompanied by uncertainty levels, there may, in certain cases, be an opportunity to rank candidates based on uncertainty. We call this Probability-of-Relevance Ranking (PRR) and we will present the exact details

in later sections. But, in overview, the approach requires the use of a relevance threshold, $\omega \in S$. When a rating is greater than, or equal to, this threshold, then the item is deemed to be relevant to the user. In the datasets we use in Section 3.4, which use a 5-star rating scale, ω might be 4, for example. In PRR, we must estimate the probability that $r_{ui} \geq \omega$, $\mathbb{P}(r_{ui} \geq \omega)$ — how certain we are that the item is relevant. Then, we rank the candidates in descending order of the relevance probabilities, so that those whose relevance is most certain are highest in the ranking. Finally, PRR recommends the top-n from this ranking.

Probability-of-Relevance Ranking was first proposed in [OLCGPB21], specifically for their BeMF method, which we will review in Section 3.2.5. Nevertheless, it also applies to many other models, as long as they are able to estimate $\mathbb{P}(r_{ui} \geq \omega)$. Some of the other methods that we review in Section 3.2 will be able to compute such probabilities by default. For some of those that cannot compute probabilities by default, we have devised a way of enabling some of them to nevertheless compute probabilities, and this is described in Section 3.5.3.3.

Notice that, with PRR, $\mathbb{P}(r_{ui} \geq \omega)$ is used to rank the items. If one also wishes to associate each recommended item with a numeric uncertainty estimate (e.g. for display to the user), then the complementary probability $1 - \mathbb{P}(r_{ui} \geq \omega)$ can be used.

3.2 Estimating Uncertainty

There are many ways of estimating the level of uncertainty of a predicted rating in an ERSs. Different sources of uncertainty can motivate different ways of estimating the uncertainty. For the purposes of surveying the approaches, we classify them into five broad types, as follows:

Information-based: Uncertainty is estimated from the amount of information that is known about users or items, or the dispersion of the ratings.

Stability-based: Uncertainty is estimated from the stability of the predictions across perturbations of the conditions under which we train the prediction model.

Error-based: Uncertainty is estimated from the expected prediction errors of an underlying rating prediction model.

Distribution-based: The ratings are assumed to follow a statistical distribution, whose dispersion is used to estimate uncertainty.

Multinomial-based: Multinomial methods also estimate uncertainty from the dispersion of the predicted ratings but they use models that make discrete predictions for each rating value.

In the following sections, we describe each of these classes in more detail; we review work from the literature that falls into each class; and we even propose new stability-based and error-based methods.

3.2.1 Information-based uncertainty

The foremost source of uncertainty in a RS comes from the lack of user-item interaction data. In practice, item catalogs are often large and therefore users will interact with only a small proportion of the items in it. In our setting, this means that users will have rated only a small proportion of the items: in the majority of cases, $\#R^{-} \gg \#R$. Models learned from this sparse data may be unreliable. For this reason, simple statistics such as item support (the number of ratings an item has) can be used to measure the reliability of the predictions [Maz13]. Hence, its negative can be used as an estimate of uncertainty:

NEG-ITEM-SUPPORT: $G_{ui} = -\#R_{\cdot i}$

Although item support is indicative of the information available about an item, it might also be misleading. For instance, two items might have the same number of ratings, but one of them might be rated similarly by different users, whereas the other might have received more divergent ratings. We expect a higher degree of uncertainty when predicting the ratings of items that have divergent ratings. For this reason, the variance of an item's ratings can also be used as an estimate of uncertainty [AKK07, Maz13]:

ITEM-VARIANCE: $G_{ui} = Var(R_{\cdot i})$

In a similar vein, we can estimate uncertainty from user profile length (the number of ratings a user has) [Maz13] or the variance of a user's ratings [WWSY13]:

NEG-USER-SUPPORT: $G_{ui} = -\#R_{u\cdot}$

USER-VARIANCE: $G_{ui} = Var(R_{u\cdot})$

Information-based measures of uncertainty can be insightful. Nevertheless, they are quite simple. One weakness is that they are model-independent, because they are computed strictly from the data, independently of the rating prediction model. In other words, changing from one prediction model to another does not result in any change to the estimates of uncertainty. This is counter-intuitive, because

different models utilise the data differently, and we would expect uncertainty estimates to reflect this behaviour. This will not be the case for information-based measures.

A second weakness is that NEG-ITEM-SUPPORT and ITEM-VARIANCE quantify uncertainty item-wise and not interaction-wise. That is, their estimated uncertainty for an item i is not personalised: $G_{ui} = \rho_{vi} \forall (u, v) \in U^2$. Similarly, NEG-USER-SUPPORT and USER-VARIANCE are strictly user-wise measures: $G_{ui} = \rho_{uj} \forall (i, j) \in I^2$.

Used in isolation, item-wise and user-wise measures lack the granularity that RSs typically need. This is especially the case for user-wise measures. A user-wise uncertainty estimate cannot help a system decide which of a set of recommendations need explanations, for example, since all of a user's recommendations will have the same uncertainty level. Similarly, user-wise uncertainty estimates cannot be used in top-n recommendations that are uncertainty-aware, such as UBF and PRR, again because the estimated uncertainty associated with each of that user's predicted ratings will be the same. In fact, since top-n recommendation is a more important task than rating prediction and since top-n recommendation is a user-based task, user-wise uncertainty estimates are largely useless. For that reason, the literature on information-based uncertainty is focused on item-wise metrics [Maz13]. In our experiments (Section 3.4), we do not include user-wise uncertainty estimates.

Nevertheless, user-wise and item-wise estimates of uncertainty can provide insights into RS performance. For example, [BFDC19] show both theoretically and empirically how eigenvalues obtained from an item-item similarity matrix are strongly related to the model recommendation accuracy. The negatives of these eigenvalues can also be interpreted as user-wise uncertainty metrics.

Finally, we note that, instead of using these estimates in isolation, some authors have proposed heuristic combinations of NEG-USER-SUPPORT and NEG-ITEM-SUPPORT (e.g. based on their product) to give interaction-wise measures (e.g. [ZTZX14, dCMC18]).

3.2.2 Stability-based uncertainty

Prediction uncertainty is related to the level of arbitrariness present in the predictions. Hence, a prediction is uncertain the more it exhibits high variance across small perturbations to the conditions under which the prediction model is

learned.

[Maz13] explores this idea in a multi-modelling strategy. Consider that a model F is used to predict ratings. Then, model stability can be measured by learning several different models from perturbations of the training dataset. A stochastic perturbation function, $\mathcal{P}$, is used to create N different versions of the original ratings training data, i.e. $\mathcal{R}^{(k)} = \mathcal{P}(\mathcal{R})$, for $k \in 1, \ldots, N$. A rating prediction model $F^{(k)}$ is learned from each of the versions of the dataset. Considering the predicted ratings as random variables, their standard deviation is an estimate of that rating's uncertainty.

In a strategy called RESAMPLE, [Maz13] uses random sampling as the perturbation function. In this case, each individual model is trained using a random subset of the ratings available for training. The predicted rating $\hat{r}_{ui}$ is given by a model F trained on the entire training set; the estimated uncertainty of that prediction is given by the standard deviation of the predictions from the models $F^{(k)}$ that are trained on the different samplings of the training set:

$$\hat{r}_{ui} = F(u, i) \tag{3.1}$$

$$G_{ui} = \frac{1}{N}\sqrt{\sum_{k=1}^{N}(F^{(k)}(u, i) - \hat{r}_{ui})^2} \tag{3.2}$$

[Maz13] proposes another such strategy, called INJECT. In this strategy, the perturbed datasets are obtained by adding noise to the original ratings, i.e. new ratings, r'_{ui} are obtained as $r_{ui} + \epsilon$, where ϵ is random Gaussian noise. The empirical results in [Maz13] show that RESAMPLE generates better estimates of uncertainty compared with INJECT, which is why, in our own empirical work (Section 3.4), we include RESAMPLE and not INJECT.

Unlike the information-based uncertainty measures, which are model-independent, the stability-based measures do depend on the prediction algorithm itself. Furthermore, the stability-based measures are interaction-wise measures. On the other hand, in order to obtain good estimates for the predicted rating variance, several different models have to be learned, hence, computing these measures can be slow. In [Maz13], this problem motivates FAST-RESAMPLE, which approximates RESAMPLE by using in each $F^{(k)}$ some model parameters that have been computed once on the original ratings.

In this dissertation, we introduce a perturbation strategy that has not previously been evaluated in the context of uncertainty estimation. It is model-specific and,

indeed, depends on using models whose parameters require random initialisation. The strategy, which we designate ENSEMBLE, consists in multi-modelling for different initialisation points. The predicted rating is the mean of the predicted ratings of the individual models, and the estimated uncertainty is their standard deviation as before:

$$\hat{r}_{ui} = \frac{1}{N} \sum_{k=1}^{N} F^{(k)}(u, i) \tag{3.3}$$

$$G_{ui} = \frac{1}{N} \sqrt{\sum_{k=1}^{N} (F^{(k)}(u, i) - \hat{r}_{ui})^2} \tag{3.4}$$

Using different initialisation points for the perturbations in multi-modelling is interesting since it means that all of the training ratings are used to learn each model $F^{(k)}$; by contrast, RESAMPLE uses only a subset of the training data for each model. This might lead to better rating estimation, and we see some evidence of this in the results of our experiments.

For our empirical work (Section 3.4), the models in the ensemble use matrix factorization (MF) (Eq. 2.1). MF is known to suffer from strong instability, depending on the initialisation of the vectors. This has been shown by [DGBC22] in a recommendation context and also by [BMNG18] and [POMT+20] for topic modelling. Ensembles of MF models that have been initialised differently can result in lower error and greater stability but also enable us to estimate uncertainty.

3.2.3 Error-based uncertainty

Uncertainty in rating predictions can be estimated through the expected error in those predictions. The more certain a prediction is, the more accurate the prediction should be. [ZOBG18] give a method for estimating prediction error, and then use this as the measure of uncertainty.

Their method use a cross-validation procedure to obtain a set of prediction errors E. First, they partition the training ratings R into K folds. It follows that each known rating $r_{ui} \in R$ will appear in exactly one fold. Then, they learn K models, $F^{(k)}$ for $k \in 1, \ldots, K$, where $F^{(k)}$ is learned from all the training ratings except those in the k-th fold. Now, they can make a prediction $\hat{r}_{ui}$ that corresponds to each known rating $r_{ui} \in R$. Specifically, if r_{ui} is in the k-th fold, then $F^{(k)}$ predicts $\hat{r}_{ui}$. This allows a calculation of training error, $e_{ui} = r_{ui} - \hat{r}_{ui}$. At the end of this cross-validation phase, they have a set of prediction errors $e_{ui} \in E$

for each $r_{ui} \in R$. Finally, they learn two models. One is a ratings prediction model F, learned from the entire ratings training set R. The second is an error prediction model, $\mathcal{E} : U \times I \to \mathbb{R}^+$, that can predict the rating prediction errors, which is learned from E.

At prediction time, we want to compute uncertain predictions, $\langle \hat{r}_{ui}, G_{ui} \rangle$. The predicted rating, $\hat{r}_{ui}$, is given by the final rating prediction model, F; the uncertainty level, G_{ui}, is the estimated prediction error, given by model $\mathcal{E}$.

In [ZOBG18], FunkSVD was chosen both as F and $\mathcal{E}$. That is, one FunkSVD model F is responsible for the rating predictions, while another FunkSVD model $\mathcal{E}$ estimates uncertainty (i.e. errors). In Section 3.4, we designate this strategy by EB-FunkSVD. More precisely, in EB-FunkSVD, we have:

$$\hat{r}_{ui} = p_u^T q_i \tag{3.5}$$

$$G_{ui} = {p'_u}^T q'_i \tag{3.6}$$

where p_u and q_i are embeddings learned from R and p'_u and q'_i are embeddings learned from E.

Nevertheless, FunkSVD is only one of many possible uncertainty estimators $\mathcal{E}$. During our empirical study, we found EB-FunkSVD to perform below our expectations. For some of our evaluation metrics, much simpler estimates, such as NEG-ITEM-SUPPORT, had better performance. These results raised a concern about possible over-complexity of FunkSVD for the uncertainty estimator. For this reason, we also experimented with a much simpler choice of $\mathcal{E}$, which we refer to as EB-Linear.

In EB-Linear, we define linear weights $b_u \in \mathbb{R}$ for $u \in U$ and $b_i \in \mathbb{R}$ for $i \in I$ for every user and item. Now, we propose that the uncertainty for a user-item pair (i.e. the predicted error) to be simply the sum of the user's and item's weights. With this formulation, instead of $\mathcal{E}$ having a d-dimensional embedding to model each user and item, there is only a single scalar parameter for each. More precisely, in EB-Linear, we have:

$$G_{ui} = b_u + b_i \tag{3.7}$$

The user and item weights, b_u and b_i, are learned, by gradient descent, to minimize the mean squared error between the predicted and observed errors.

Finally, we also acknowledge the contribution by [CTFLHT13], where another uncertainty estimation approach based on prediction errors was introduced. There,

uncertainty estimation is viewed as a binary classification task, where the goal is to segregate certain from uncertain predictions. This task is solved by learning a classifier that predicts whether a user-item interaction is uncertain or not. The classifier is learned based on both the prediction errors committed by the rating prediction model (in their case, a k-nearest neighbours model) and the ratings given by similar users to the same item. We will not further explore this approach for two reasons. First, it is restricted to neighborhood models. Second, the segregation of items into two types (certain and uncertain) is quite different from the formulation used in our work, where we seek to estimate the level of uncertainty, G_{ui} (Section 3.1). In any case, EB-FunkSVD and EB-Linear are also based on prediction errors and can be applied to a wider class of collaborative filtering models.

3.2.4 Distribution-based uncertainty

The RS literature contains several methods that consider each user-item rating to be a random variable, each having a certain probabilistic distribution $\mathcal{F}|\Theta_{ui}$, where Θ is the set of distribution parameters. Modelling the ratings with a distribution rather than a point estimate allows the recommender to produce a much richer output, from which it is possible not only to obtain point estimates for the ratings but also to infer the uncertainty associated with these point estimates.

The Gaussian distribution emerges as a natural choice: its mean, μ, and variance, σ^2, are obvious choices for the rating point estimates and their uncertainties. In this case, $r_{ui} \sim \mathcal{N}(\mu_{ui}, \sigma^2_{ui})$, and the task of estimating r_{ui} is performed by estimating $\Theta_{ui} = \langle \mu_{ui}, \sigma^2_{ui} \rangle$ for every user-item pair.

One notable algorithm to employ a Gaussian model is Probabilistic Matrix Factorization (PMF) [MS08]. In PMF, the ratings are assumed to follow a Gaussian distribution, with a fixed variance parameter $\sigma^2 = 1$. As before, let p_u and q_i be user and item embeddings, both of dimension d. Then, the likelihood of the training data can be written as:

$$\mathbb{P}(R) = \prod_{i \in I} \prod_{u \in U} \left[\mathcal{N}(r_{ui} | p_u^T q_i, 1) \right]^{\delta_{ui}} \tag{3.8}$$

where $\mathcal{N}(\cdot|\mu, \sigma^2)$ is the probability density function of the Gaussian distribution with mean μ and variance σ^2, and δ_{ui} is an indicator function that is equal to 1 if user u rated item i and 0 otherwise. The latent factors are assumed, in prior, to follow a zero-mean spherical Gaussian distribution. In this case, maximising the

log-posterior distribution is equivalent to minimising the sum of squared errors subject to L_2 regularisation terms:

$$\sum_{u \in U} \sum_{i \in I} \delta_{ui} (r_{ui} - p_u^T q_i)^2 + \lambda_U \sum_{u \in U} ||p_u|| + \lambda_I \sum_{i \in I} ||q_i|| \tag{3.9}$$

To learn PMF from explicit ratings, we can use mini-batch gradient descent, iterating through the observed rating. The procedure is equivalent to the one described in Section 2.3.2. The loss function of a batch of ratings B is,

$$\mathcal{L}(\mathcal{B}|\theta) = \frac{1}{\#\mathcal{B}} \sum_{(u,i,r_{ui}) \in \mathcal{B}} (r_{ui} - p_u^T q_i)^2 + \lambda_U \sum_{u \in U} ||p_u|| + \lambda_I \sum_{i \in I} ||q_i|| \tag{3.10}$$

Notice that, although having a different motivation, the PMF model optimizes an objective function which is similar[2] to FunkSVD's (Eq. 2.7).

PMF assumes every user-item pair to have the same variance. Not only is this a very strong assumption in general, it also means that PMF is not suitable for uncertainty estimation, since every user-item pair would have the same estimated uncertainty σ^2. For this reason, [WLW+18] extend PMF to Confidence-aware Probabilistic Matrix Factorization (CPMF). There, each user and item is assumed to have a latent variance parameter, σ_u and σ_i, respectively. The variance of each user-item pair is obtained by some function $\gamma : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^+$ that composes these two latent variance parameters. Therefore, the likelihood of the training data is:

$$r_{ui} \sim \mathcal{N}(p_u^T q_i, \gamma(\sigma_u, \sigma_i)) \tag{3.11}$$

The latent factors, p_u and q_i, and the variance parameters, σ_u and σ_i, are estimated through gradient descent on the negative log-likelihood.

[WLW+18] use a simple product for γ. In our experiments (Sections 3.4), we do the same and this is what we are referring to when we write CPMF in that later section of this chapter.

[WLW+18] also propose a Bayesian version of the algorithm, similar to [SM08]. The idea of the Bayesian framework is to automatically control model complexity through prior distributions in order to alleviate the effects of overfitting and to

[2] In fact, user and item specific regularization parameters λ_U, λ_I, may also be used with FunkSVD, making it equivalent to PMF.

improve model generalisation.

Another distributional method that resembles CPMF is proposed by [YB21]. Their proposal, which is based on Gaussian Processes, also assumes each interaction to follow a Gaussian distribution. However, their method explicitly models the full covariance between all pairs of interactions. For this reason, their model is more complete then CPMF, but also a lot slower to train, especially for big datasets.

The next section presents further distribution-based methods for the special case where ratings are not treated as continuous variables.

3.2.5 Multinomial-based uncertainty

Ratings are typically discrete; for example, the ratings in the Netflix Prize dataset are integers from 1 to 5 [BL07]. But, in rating prediction, ratings are often treated as continuous variables. In other words, the model predicts real values: $F : U \times I \to \mathbb{R}$. Then, as we have seen, there are various ways of augmenting each predicted rating with an estimated uncertainty level.

Multinomial models treat predicted ratings as discrete variables. For each point on the rating scale, they predict a probability. Hence, the output of a multinomial prediction model is a vector, containing one probability for each score s on the rating scale $S = \{s_1, \ldots, s_m\}$, where $\mathbb{P}(r_{ui} = s)$ is the probability that user u will assign rating $s \in S$ to item i, and $\sum_{s \in S} \mathbb{P}(r_{ui} = s) = 1$. For instance, in the Netflix case, the model will predict $\mathbb{P}(r_{ui} = s)\, \forall s \in \{1, \ldots, 5\}$. This idea has motivated several recommendation algorithms, including the User Rating Profile Model (URP) [Mar03], Bernoulli Matrix Factorization (BeMF) [OLCGPB21] and OrdRec [KS11]. From a statistical standpoint, this way of formulating rating prediction consists of assuming r_{ui} follows a multinomial distribution. Hence, it is possible to extract not only point estimates for the ratings, but also dispersion statistics, which can be used as estimates of uncertainty.

The BeMF model is one way to estimate the vector of probabilities [OLCGPB21]. BeMF learns m different Bernoulli factorization models, one for each point on the rating scale, i.e. it learns a set of models $F = \{F_1, \ldots, F_m\}$. To learn each individual model F_s, it uses a modified version of the training set $R_s = \{(u, i, r_{uis})\}$, such that:

$$r_{uis} = \begin{cases} 1, & r_{ui} = s \\ 0, & \text{otherwise} \end{cases} \quad \forall (u, i, r_{ui}) \in R \tag{3.12}$$

Each Bernoulli factorization model F_s makes predictions in the same way as the FunkSVD method (Equation 2.1), but with the difference that the predictions are scaled through a logistic function, which bounds them to the $[0, 1]$ interval. To learn the latent factors, for each individual model, the cross-entropy loss function is used. The loss for a single training instance is:

$$F_s(u,i)^{r_{uis}}(1 - F_s(u,i))^{1-r_{uis}} \tag{3.13}$$

The parameters of the model are updated by performing gradient descent on this loss function. Finally, the probabilities are given by:

$$\mathbb{P}(r_{ui} = s) = \frac{F_s(u,i)}{\sum_{s \in S} F_s(u,i)} \tag{3.14}$$

With the probabilities estimated, [OLCGPB21] proposes that the predicted rating will be the most probable value in the rating scale, and the uncertainty estimate will be the probability mass lying outside this most probable rating value:

$$\hat{r}_{ui} = \arg_{s \in S} \max \mathbb{P}(r_{ui} = s) \tag{3.15}$$

$$G_{ui} = 1 - \max_{s \in S} \mathbb{P}(r_{ui} = s) \tag{3.16}$$

A rather different multinomial approach, OrdRec, is proposed in [KS11]. While OrdRec has a very different formulation from BeMF, its ultimate goal is also to predict the vector of probabilities. OrdRec learns a single rating prediction model, e.g. using FunkSVD, which will be denoted F. Then, for every $s \in \{s_1, \ldots, s_{m-1}\}$, the cumulative probability function for the ratings is:

$$\mathbb{P}(r_{ui} \leq s) = \frac{1}{1 + e^{F(u,i) - t_{us}}} \tag{3.17}$$

where t_{us} for each $s \in S$ is a user-specific set of threshold parameters. Notice that $\mathbb{P}(r_{ui} \leq s_m) = 1$ as this is a cumulative distribution. After estimating the cumulative functions, the individual probabilities are obtained by:

$$\mathbb{P}(r_{ui} = s_i) = \mathbb{P}(r_{ui} \leq s_i) - \mathbb{P}(r_{ui} \leq s_{i-1}) \tag{3.18}$$

Now, for every $u \in U$, let $t_{u0} = 0$. Then, the gaps between a user's thresholds

are encoded as a set of parameters $\beta_{u1}, \beta_{u2}, \ldots, \beta_{um-1}$, such that

$$t_{us_i} = t_{us_{i-1}} + e^{\beta_{us_i}}, \quad \forall i \in \{1, \ldots, m-1\} \tag{3.19}$$

Therefore, the whole set of parameters to be learned for OrdRec includes the underlying model's parameters, as well as $\{\beta_{u1}, \beta_{u2}, \ldots, \beta_{um-1}\} \forall u \in U$. The likelihood function for the model is:

$$\prod_{i \in I} \prod_{u \in U} \prod_{s \in S} (\mathbb{P}(r_{ui} = s))^{\delta(r_{ui}=s)} \tag{3.20}$$

Learning is performed by stochastic gradient descent on the negative log-likelihood function.

With the rating distribution estimated, there are many ways in which rating predictions and uncertainty estimates can be extracted. [KS11] argue that the distribution's average and some measure of its dispersion can be used to predict ratings and uncertainty respectively. Moreover, any distribution dispersion metric can be employed, such as standard deviation, Gini impurity or entropy. Among these, they empirically found that the standard deviation performed best for uncertainty and, therefore, we will use the same when we estimate uncertainty using OrdRec in this chapter:

$$\hat{r}_{ui} = \sum_{s \in S} s\, \mathbb{P}(r_{ui} = s) \tag{3.21}$$

$$G_{ui} = \sqrt{\frac{1}{m} \sum_{s \in S} (s^2 \mathbb{P}(r_{ui} = s)) - m\hat{r}_{ui}^2} \tag{3.22}$$

Moreover, we highlight one concern regarding the uncertainty estimates given by Equation 3.22: when $\hat{r}_{ui}$ is either too small or too big, that is, close to s_1 or to s_m, then the uncertainty G_{ui} will always be very low. This happens because, in order to predict one of the distribution's extremes, the vast majority of the distribution's probability have to fall on this extreme rating value, leading to very low standard deviation. Intermediate rating values do not suffer from the same problem because, in this case, the distribution's probability can be dispersed symmetrically around that intermediate rating value.

We have concluded our review of ways of *estimating* ratings prediction uncertainty. We turn our attention now to ways of *evaluating* these estimates.

3.3 Evaluating Uncertainty Estimates

After training a rating prediction model, it is common to evaluate its performance on a test set, this being a set of known ratings that was held-out during training. The most common rating prediction evaluation metrics, such as root mean squared error (RMSE), simply compare the predicted ratings to the known ratings in the test set (see Eq. 2.15).

In this case, the known ratings give us a 'ground-truth' for the evaluation. Similarly, after estimating the uncertainty of predicted ratings, it is desirable to evaluate the quality of the uncertainty estimates. But, the evaluation of the uncertainty estimates is not as simple as evaluating the predicted ratings because there are no 'ground-truth' values that can be compared with the uncertainty estimates.

Our review of ways of evaluating estimates of rating prediction uncertainty starts with a discussion of expected correlations. Subsequent sections then correspond to the two different tasks that an ERS perform: rating prediction and top-n recommendation, the latter being split into Rating-Based Ranking, Uncertainty-Based Filtering and Probability-of-Relevance Ranking.

3.3.1 Expected correlations

Despite the differences in the uncertainty estimates that we reviewed in Section 3.2, there are 'patterns' that we expect to observe, including the following:

Negative correlation with user profile size: The less we know about a user's tastes, the more uncertain we expect that user's predictions ratings to be.

Negative correlation with item support: Similarly, the less that an item has been interacted with, the less certain we expect that item's predicted ratings to be.

Positive correlation with user rating variance: The more a user's ratings vary from each other, the less certain we expect that user's predictions to be.

Positive correlation with item rating variance: The more an item's ratings vary from each other, the less certain we expect that item's predictions to be.

To the best of our knowledge, we are the first to propose checking these correlations. Computing them can serve as an initial check that a particular way of estimating uncertainty behaves, on the whole, in the way we might expect. We have chosen to carry out this check using Spearman rank correlation, because the correlations might not always be linear.

It is important to point out that some of these correlations might not always hold for individual user-item interactions. For example, in the movie domain, a new movie from a famous director is usually a very safe and expected recommendation, even though the movie, being newly-released, will have low item support.

We also expect to see some correlations between the uncertainty estimates and prediction error and top-n recommendations accuracy:

Prediction error: The more wrong a predicted rating is, the less certain we expect the prediction to be.

Recommendation accuracy: Similarly, the more accurate a set of top-n recommendations is, the more certain we expect the predicted rating for each item in the top-n to be.

The literature offers several metrics that we can compute to confirm these correlations, and we explore these in more detail in the following subsections.

3.3.2 Rating prediction

Uncertainty measures are expected to reflect prediction errors. For this reason, the foremost method for evaluating the quality of estimates of rating prediction uncertainty consists of simply evaluating the correlation between the prediction errors in a holdout test set and their estimated uncertainty. Different correlation coefficients can be used. For example, Pearson correlation can be used to measure the linear correlation, while Spearman correlation can be applied if rank correlation is preferred. Later, we refer to these correlations (between rating prediction errors and uncertainty estimates) as Pearson_ρ and Spearman_ρ.

Mazurowski and Maciej [Maz13] proposes a visualization of the correlation between uncertainty and prediction error. In fact, they refer to "reliability estimates" on predicted ratings, rather than uncertainty estimates. But, as their concept of reliability is the opposite of uncertainty, their visualisation can be used here, after some adaptations. In what follows, we 'translate' their proposals to our uncertainty framework.

Their proposal begins by splitting the predictions for the test set into B equal-sized bins based on, and ordered by, their uncertainty levels — that is, bin 1 contains the least uncertain predictions and bin B the most uncertain ones. For each bin b, an error threshold, ϵ_b is calculated. The error threshold for bin b measures how inaccurate the predicted ratings within the bin are. The thresholds are calculated such that Bayesian confidence intervals of the form $[\hat{r}_{ui} - \epsilon_b, \hat{r}_{ui} + \epsilon_b]$ contain the real rating r_{ui} for at least $(1 - \alpha) \times 100\%$ of the bin's predictions, where $\alpha \in [0, 1]$ is a fixed significance level. The quality of the confidence intervals are determined by the curve of the error thresholds against their confidence bin.

One problem with the latter method is that the curves will depend on the choice of α. In fact, the reliance on α means that each uncertainty estimate will have a family of curves (for each value of α), which makes it more difficult to compare uncertainty estimates. To solve this issue, and also to adapt Mazurowski and Maciej [Maz13] confidence intervals to our uncertainty framework, we propose to use uncertainty bins, rather than confidence bins, and to calculate the average RMSE in each of these uncertainty bins. In this case, given B equal-sized uncertainty bins, we calculate the following for each bin b:

$$RMSE_b = \sqrt{\frac{\sum_{G_{ui} \in b} (\hat{r}_{ui} - r_{ui})^2}{\#G_{ui} \in b}} \tag{3.23}$$

where the denominator is the number of test instances falling into bin b. Similarly to before, we then plot the curve of the $RMSE_b$ values against their bin index. We refer to these as RMSE-uncertainty curves. Now these curves have no reliance on additional parameters other than the number of reliability bins — in particular, there is no longer the parameter α.

Nevertheless, a problem that remains is how to interpret curves like these. To illustrate, we refer to Figure 3.1. The Figure shows three example RMSE-uncertainty curves. Notice that all the curves have very similar $RMSE_1$ and $RMSE_B$ values. On the other hand, their behaviours in between these extremes are different. In this case, there is no clear definition that states which curve is preferred, and hence which of the underlying uncertainty estimates are better.

To alleviate this problem, some simple metrics can be extracted from the curves. One option, which has been explored in [Maz13], is to calculate the difference in the values between the first and last bins. In [Maz13], this metric is called d_w. Here, to avoid confusion due to the fact that we are using RMSE-uncertainty curves, we will refer to it as $\delta RMSE$:

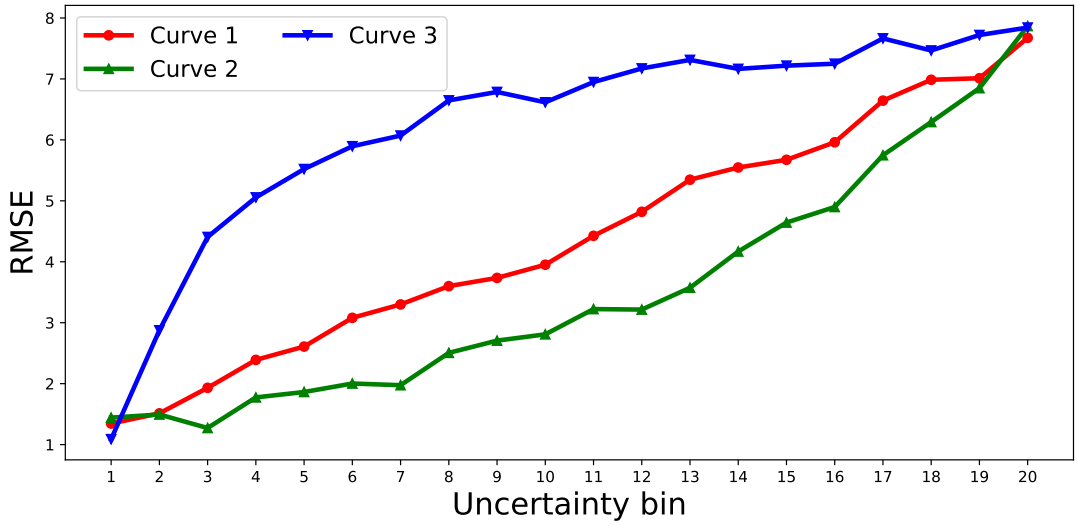


Figure 3.1: Example RMSE-Uncertainty curves.

Table 3.1: Criteria for a good estimate of rating prediction uncertainty

Prediction error	Uncertainty	Uncertainty evaluation metric
High	Low	Big penalty
Low	High	Small penalty
High	High	Big reward
Low	Low	Small reward

$$\delta RMSE = RMSE_B - RMSE_1 \tag{3.24}$$

Higher values of $\delta RMSE$ are indicative that prediction error grows as uncertainty grows.

In a similar vein, [BGOZ18] argue that an evaluation metric for an uncertainty measure should penalise predictions where certainty is high but predictive error is also high and should strongly reward predictions where uncertainty is high and predictive error is high. Table 3.1 schematically describes the desired behaviour.

[BGOZ18] propose a metric that matches the criteria in Table 3.1. Like the work in [Maz13], Bobadilla et al.'s Reliability Prediction Improvement metric is given in terms of reliability. We adapt it from reliability to uncertainty and call it Uncertainty Prediction Improvement (UPI).

To define UPI, we first compute the absolute errors $e_{ui} = |\hat{r}_{ui} - r_{ui}|, \forall (u, i, r_{ui}) \in T$,

where T is a holdout test set. Then, we define the metric criteria, C^{UPI}, as follows:

$$C_{ui}^{UPI} = e_{ui}(e_{ui} - \overline{e})(\overline{\rho} - G_{ui}) \tag{3.25}$$

where $\overline{e}$ and $\overline{\rho}$ are the means of the absolute errors and the uncertainty estimates for all the test instances in T. UPI is then defined as:

$$UPI = \frac{\frac{1}{\sigma_e \sigma_\rho \# T} \sum_{r_{ui} \in T} C_{ui}^{UPI}}{\overline{e}} \tag{3.26}$$

where σ_e and σ_ρ are the standard deviations of the absolute errors and uncertainty estimates. The divisor term, $\sigma_e \sigma_\rho \# T$, standardises the criteria values, while the term $\overline{e}$ compares the standardised criteria values to the mean absolute error.

[KS11] take a rather different approach to the evaluation of uncertainty estimates. Instead of using metrics, such as $\delta RMSE$ and UPI, they build and evaluate what they call *confidence classifiers*. These are binary classifiers that use the uncertainty estimates as predictors (features) to predict whether or not the rating error will be greater than 1. In practice, given a set of predicted ratings $\hat{r}_{ui}$ and their respective observed ratings r_{ui}, a confidence classifier is a logistic model, where the uncertainty estimate is the predictive feature, and the model predicts the probability $\mathbb{P}(|\hat{r}_{ui} - r_{ui}| > 1)$. Once trained, the confidence classifier is evaluated using a holdout set. Its accuracy on the holdout set is an indicator of the effectiveness of the uncertainty estimate. Koren and Sill [KS11] use the classifier's Area Under the Curve (AUC) for this.

Care must be taken to avoid training and evaluating the confidence classifier on the instances used to train the rating predictor. In our experiments, we train the rating predictor on the training set, then we use 2-fold cross-validation on the test set to train and evaluate the confidence classifier. In other words, we train the confidence classifier on half of the test instances and evaluate its AUC on the other half of the test instances. We repeat this, after interchanging the two folds. The final results that we report are the average AUC across the two folds. In Section 3.5, we refer to this approach to evaluation as Error-Uncertainty Classification (EUC).

We highlight that the choice to predict $\mathbb{P}(|\hat{r}_{ui} - r_{ui}| > 1)$ with this method is arbitrary. The method is general: we could build confidence classifiers based on thresholds other than 1. In our experiments, however, we follow [KS11]'s use of 1.

3.3.3 Top-n recommendation

We divide our discussion of ways of evaluating the quality of the uncertainty estimates at top-n this into three. First, we discuss the evaluation of uncertainty in recommendations given with RaBR. Then, corresponding with the use cases introduces in Section 3.1, we propose an application-based approach, where we access the validity of the uncertainty estimates according to the improvements they provide to these use cases.

3.3.3.1 Rating-based ranking

As previously stated, we expect recommendation accuracy to be greater in cases where the recommendations are less uncertain. Therefore, one metric to evaluate the uncertainty estimates is the Spearman correlation between some recommendation accuracy metric, e.g. MAP@n, and the average uncertainty of the recommended items for the user. We will refer to this as Uncertainty Accuracy Correlation (UAC). Notice that, for this metric, smaller values denote better performance.

In a similar vein, [BGOZ18] propose Reliability Recommendation Improvement, a metric that evaluates reliability estimates in the case of recommendations. Their metric is based on similar criteria to those in Table 3.1, this time that hits (top-n recommendations that are correct) should be associated with lower uncertainty levels than the non-hits. We present Uncertainty Recommendation Improvement (URI), which is our adaptation from reliability to uncertainty.

Our adaptation starts by defining C^{URI} as follows:

$$C_u^{URI} = \sum_{i \in Z_u^n | r_{ui} \geq \omega} (\overline{\rho} - G_{ui}) \tag{3.27}$$

Notice that there is a strong relation with C^{UPI}. However, URI is focused on relevant items, i.e. test set ratings for which $r_{ui} \geq \omega$. Hence, in C_u^{URI}, the mean uncertainty $\overline{\rho}$ is calculated only over test instances where $r_{ui} \geq \omega$.

In this definition of C^{URI}, based on [BGOZ18] and similar to C^{UPI}, $\overline{\rho}$ is a global average — the average of all the uncertainty values in test set T. In other words, it is not the average of these values just for user u. We believe that averaging over all T can lead to problems. For example, it is likely that some users will have high overall uncertainty. In an extreme case, there may even be users u for which $G_{ui} < \overline{\rho} \quad \forall i \in I$. For such users, if a recommendation hit occurs, then $C_u^{URI} < 0$

even if the recommended hit is less uncertain than all of the other recommended items for that user. Given a set of recommendations for a user, it is desirable for recommendation hits to have a lower estimated uncertainty than non-hits. For this reason, we propose an alternative definition of URI that is based on Z_u^n, the top-n items recommended to u. We redefine it as follows:

$$C_u^{URI} = \sum_{i \in Z_u^n | r_{ui} \geq \omega} (\overline{\rho}_u - G_{ui}) \tag{3.28}$$

where $\overline{\rho}_u$ is the mean uncertainty for the items in Z_u^n, i.e. the n items that are recommended to user u.

Now, the URI is defined as:

$$URI = \frac{\sum_u C_u^{URI}}{\sigma_{\rho_u} \sum_u \#\{i \in Z_u^n | r_{u,i} \geq \omega\}} \tag{3.29}$$

where σ_{ρ_u} is the standard deviation of the uncertainty values for Z_u^n. The denominator is used to guarantee that the URI is a valid metric when comparing different uncertainty measures.

3.3.3.2 Uncertainty-based filtering

In Section 3.1.2.1, we argued that, in some circumstances, it can be useful to constrain the set of items that might be included in a top-n by discarding candidates whose uncertainty value exceeds some threshold τ. As initially proposed in [OLCGPB21], we can use this as a way of evaluating uncertainty estimates. In certain cases, the recommendation accuracy (e.g. its precision) can increase — if the candidate items that are discarded are not only uncertain but also not relevant. In practice, the recommendation accuracy can be measured for a range of different values for τ, with the expectation that accuracy will increase as τ increases. Nevertheless, restricting the candidate set might also decrease the average predicted rating in the recommended items. For instance, before filtering, a user might have several candidates that receive high predicted ratings but also high uncertainty. If these are filtered out, not only will uncertainty be reduced in the filtered recommendation set, but also the average predicted rating. For this reason, together with the accuracy, it is important to monitor the Mean Predicted Rating. To do so, we use:

$$\text{Mean Predicted Rating@}n = \frac{1}{\#U} \sum_{u \in U} \frac{1}{n} \sum_{i \in Z_u^n} \hat{r}_{ui} \tag{3.30}$$

Finally, since we are preventing the RS from recommending highly uncertain items to the user, there may be users for whom the RS cannot recommend a full set of n items. Hence, the recommendation *coverage* should also be measured. In our experiments, we measure coverage according to Eq. 2.27.

3.3.3.3 Probability-of-relevance ranking

In RaBR, uncertainty estimates are not used for ranking the candidates and selecting the top-n, and therefore uncertainty in the case of RaBR has to be evaluated through specific metrics, such as UPI. In PRR, on the other hand, uncertainty estimates are used to rank the candidates and select the top-n; specifically, we recommend the candidates for which $P(r_{ui} \geq \omega)$ is highest. This means that, in the case of PRR, the uncertainty estimates can be evaluated through recommendation accuracy metrics. That is, the quality of the recommendation set given by the uncertainty estimates serve as an indirect way to evaluate these estimates.

Furthermore, in the cases where the system is expected to provide uncertainty estimates together with the uncertainty-based recommendations, then these uncertainty values also have to be evaluated. In Section 3.1.2.2, we propose that $1 - \mathbb{P}(r_{ui} \geq \omega)$ can be used as the uncertainty estimate. In this case, the uncertainty estimates will increase as we progress through the item ranking. That is, the uncertainty of the top-ranked item will be the smallest, followed by the second, and so on. Under this setup, URI is not a feasible evaluation metric anymore, because the differences $\bar{\rho}_u - G_{ui}$ will always be the largest for the top-ranked items, leading to unrealistically high URI values. On the other hand, the correlation between the uncertainty estimates and accuracy (UAC) is still a feasible metric, and we will use it below.

3.4 Empirical Study

In this section, we report the results of an empirical study that we have conducted. We believe it is the most extensive study in the field to this date. Compared with previous studies, it covers more ways of estimating uncertainty and uses more evaluation methods, as well as making use of two large datasets. Previously, the

most comprehensive study was [Maz13], but it compared only information-based and stability-based uncertainty estimates. The study in [BGOZ18] proposed and empirically tested some uncertainty quality measures but, similarly, it did not consider error-based, distribution-based or multinomial-based uncertainty estimates. Other studies in the field, e.g. [KS11, WLW+18], use few, or no, uncertainty evaluation metrics, and therefore also lack empirical support for their uncertainty estimates. Finally, [OLCGPB21] has two major issues with their experiments: it does not compare BeMF against OrdRec, which has a similar motivation to it; and it has an unconventional recommendation evaluation, where candidates for recommendation are taken only from the test set and therefore unrated items are never recommended.

Our implementation makes considerable use of PyTorch [PGC+17] and is partially inspired by the Spotlight Python libraries.[3] For reproducibility, all the code used is available.[4]

3.4.1 Models

We compare the following ways of estimating uncertainty, each of which was described in Section 3.2: NEG-ITEM-SUPPORT, ITEM-VARIANCE, RESAMPLE, ENSEMBLE, EB-FunkSVD, EB-Linear, CPMF, BeMF and OrdRec. As explained previously, we do not include the user-wise estimates, NEG-USER-SUPPORT or USER-VARIANCE: they assign the same level of uncertainty to all of a user's candidate items, which is not helpful to a top-n recommender system.

Most of these techniques rely on a separate rating predictor; only CPMF, BeMF and OrdRec have their own methods for predicting the ratings. For those that need a separate rating predictor, we use FunkSVD (Equation 2.1). Moreover, FunkSVD is also the underlying rating prediction model F in the case of OrdRec and BeMF, and has the same formula as the distribution average in the case of CPMF (Equation 3.11).

Furthermore, we also use FunkSVD separately as a baseline for recommendation accuracy. We will want to determine whether the new techniques have similar or better prediction error (RMSE) and top-n accuracy (MAP@n and Recall@n) than FunkSVD.

[3] `https://github.com/maciejkula/spotlight`
[4] `https://github.com/vcoscrato/uncertain/tree/SnapExplicit`

Table 3.2: Characteristics of the datasets used in our experiments.

Dataset	MovieLens	Netflix
num. users, $\#U$	162,542	53,424
num. items, $\#I$	59,047	480,189
num. ratings, $\#R$	25,000,095	100,480,507
sparsity	99.74%	99.61%
avg. num. ratings per user	154	1881
avg. num. ratings per item	423	209
rating scale	$\{0.5, 1, 1.5, \ldots, 5\}$	$\{1, 2, \ldots, 5\}$

3.4.2 Datasets

We use two large ratings datasets: the MovieLens 25 million ratings dataset [HK15] and the Netflix prize dataset [BL07]. Table 3.2 shows some information about each dataset. We chose these datasets because they are by far the most popular ones for the rating prediction task.

3.4.3 Methods

3.4.3.1 Data splitting

We select a random sample of 10,000 users to be test users. The restriction to 10,000 users is to reduce the computational cost of running this extensive set of experiments on such large datasets. For these sampled test users, we chronologically order their ratings and place the last 20% of them in the test set. We use the remaining ratings (from *every* user, not just the test users) for training or validation. Specifically, we order these remaining ratings chronologically; for each user, we use the first 80% of these ratings for our training set, and the rest for our validation set.

3.4.3.2 Tuning

We have a number of hyper-parameters, whose values we need to choose. We tune the models by training them with different hyper-parameter values on the training set and evaluating them on the validation set, choosing values that give the lowest validation set RMSE.

Wherever we are using matrix-factorization, the latent feature vector dimension d and the regularisation strengths λ_U and λ_I are hyper-parameters. The latent feature vector dimension is chosen from $\{50, 100, 200\}$, while the regularisation strengths are chosen from $\{0.1, 0.01, 0.001\}$. The latent feature vectors are ini-

tialised by sampling from a zero-centered Gaussian distribution with a standard deviation of 0.01. We set the learning rate to 0.0001 in all cases and used the Adam optimiser to account for learning rate adaptations. Furthermore, to suppress the need to optimise the number of training epochs, we employed early-stopping, monitoring RMSE after every epoch and stopping training when validation RMSE does not improve for five consecutive epochs. This setup is similar to the one used in [DGBC22], for example.

For ENSEMBLE and RESAMPLE, which involve multi-modelling (Eqns. 3.3 and 3.4), we use $N = 5$ different models. For RESAMPLE, we use an 80% sample fraction. For the error-based methods, EB-FunkSVD and EB-Linear, we use 2-way cross-validation to create the error matrix E. We conducted some preliminary experiments with more multi-modelling runs (10 and 20) and more CV-folds (5 and 10), but we obtained similar results to the ones for $N = 5$ and for 2-way cross-validation.

BeMF trains multiple MF models, one per rating value. For fairness, we ensure that BeMF has a similar number of parameters to all the other models. For example, for the Netflix dataset, where there are 5 possible rating values, if the other models are using $d = 50$, then BeMF will learn five models with dimension 10 each.

3.4.3.3 Rating prediction evaluation

To evaluate rating predictions, we measure the RMSE between the ratings in the test set and the predicted ratings. To evaluate the uncertainty estimates, we use the following evaluation techniques that we presented in Section 3.3.2: Pearson$_\rho$, Spearman$_\rho$, UPI, RMSE-Uncertainty curves, $\delta RMSE$ and EUC.

3.4.3.4 Top-n recommendation evaluation

We compute recommendations for each of the 10,000 test users. For each of these users u, the candidate items $C_u \subseteq I$ that can be recommended to u are all the items that u has has not rated in either the training or validation sets. Then, for each user, we create a top-n, $Z_u^n \subseteq C_u, |Z_u^n| = n$. The way we do this depends on whether we are evaluating RaBR, UBF or PRR. We measure the accuracy of a top-n using MAP@n[5] and Recall@n, averaged over all 10,000 users. An item

[5]For readability, we chose to present only one of the two position-aware recommendation accuracy metrics presented in Chapter 2. The choice for MAP over NDCG was arbitrary. From our experience, they are generally strongly correlated.

i is considered relevant to a user u if its test set rating $r_{ui} \geq \omega$ where $\omega = 4$. We evaluate the uncertainty estimates using the evaluation techniques that we presented in Section 3.3.3, namely UAC and URI.

3.5 Results

In this section, we present the results. The subsections correspond to the ones in Section 3.3.

3.5.1 Expected correlations

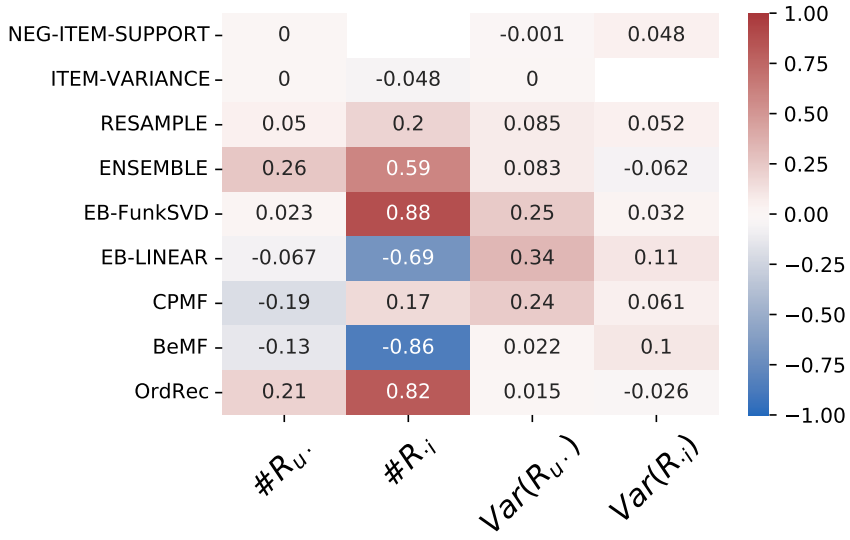
We start by exploring correlations between the uncertainty estimates and each of: the user profile size ($\#R_{u\cdot}$), item support ($\#R_{\cdot i}$), user rating variance ($Var(R_{u\cdot})$) and item rating variance ($Var(R_{\cdot i})$).

We compute Spearman correlations on a set of 100,000 randomly sampled user-item pairs. In other words, we pick a random user and a random item, we estimate the uncertainty G_{ui} and, once we have done this for 100,000 such pairs, we correlate with the statistics mentioned in the previous paragraph. Selecting users and items at random avoids a bias: if we had instead selected users and items from R ($r_{ui} \in R$), then we would bias these correlations to cases where the user has rated the item.

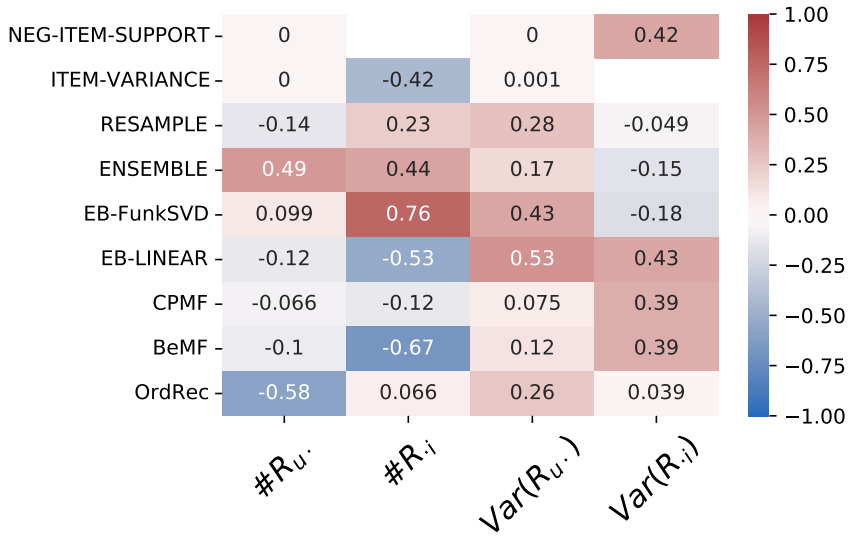
Figure 3.2 gives the results. The correlations between NEG-ITEM-SUPPORT and $\#R_{\cdot i}$ and between ITEM-VARIANCE and $Var(R_{\cdot i})$ are trivially perfect and so they are not included.

In the MovieLens dataset, the models' correlation with user profile size, user variance and item variance are very low; EB-Linear and BeMF show strong negative correlation with item support, as expected; but there are counter-intuitive positive correlations with item support for ENSEMBLE, EB-FunkSVD and OrdRec. In the Netflix dataset, some expected positive correlations with user variance and item variance can be observed, with the most noticeable values being achieved by EB-Linear; in the case of item support, EB-Linear and BeMF show strong negative correlation, as expected, while RESAMPLE, ENSEMBLE and EB-FunkSVD show positive correlations to it, which is counter-intuitive.

Notwithstanding the heuristic nature of these expected correlations, the results are quite concerning. Although they are all supposed to be estimates of uncertainty, they behave differently from each other, and there is no uncertainty



(a) MovieLens dataset.



(b) Netflix dataset.

Figure 3.2: Correlations between uncertainty estimates and dataset statistics. We expect negative correlation with $\#R_{u\cdot}$ and $\#R_{\cdot i}$; we expect positive correlation with $Var(R_{u\cdot})$ and $Var(R_{\cdot i})$.

Table 3.3: Rating prediction: RMSE. (Results for NEG-ITEM-SUPPORT, ITEM-VARIANCE, RESAMPLE, EB-FunkSVD and EB-Linear will be the same as the Baseline.) The best values (lowest) are highlighted in bold.

	RMSE	
Model	**MovieLens**	**Netflix**
Baseline	0.8400	0.8618
ENSEMBLE	**0.8294**	**0.8493**
CPMF	0.8447	0.8742
BeMF	1.0031	1.0270
OrdRec	0.8832	0.8666

estimate that convincingly exhibits all the expected behaviours. We highlight, among these concerning results, how EB-FunkSVD and EB-Linear have an opposite correlation with item support. This is particularly odd because both methods are based on the errors of the same baseline, and yet their uncertainty estimates differ drastically. While some of the forthcoming results in the reminder of this work will show good performance for these methods, the correlations we have observed raise a flag to the credibility of uncertainty estimates based on prediction error.

In the remaining subsections, we evaluate the uncertainty estimates using the methods that we reviewed in Section 3.3.

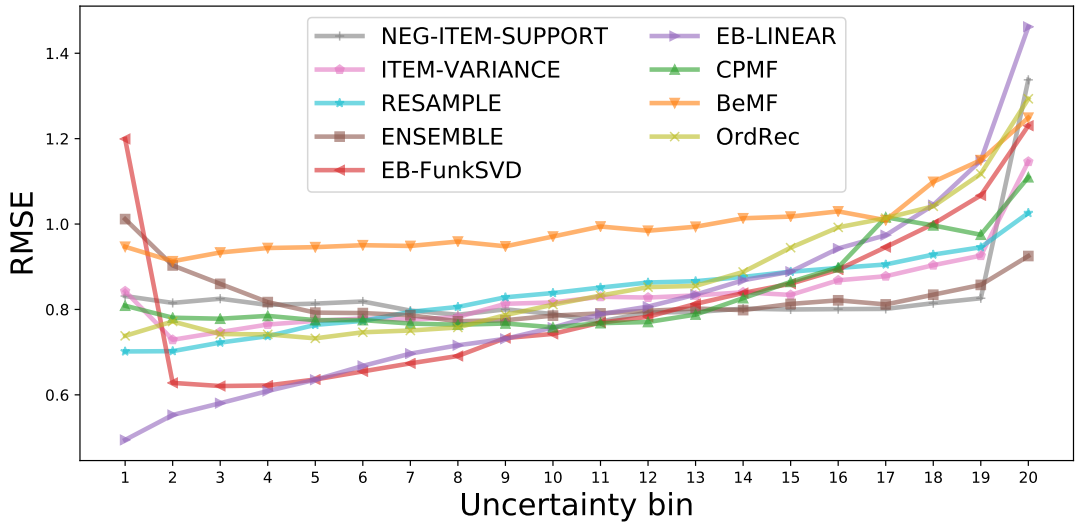
3.5.2 Rating prediction

We begin our evaluation of rating prediction by comparing the prediction error of the different models, leaving uncertainty aside. Results are shown in Table 3.3. Several methods (NEG-ITEM-SUPPORT, ITEM-VARIANCE and RESAMPLE) are not explicitly listed in this Table. This is because, for rating prediction itself, they use the baseline model, and therefore their RMSE is the same as that of the baseline.

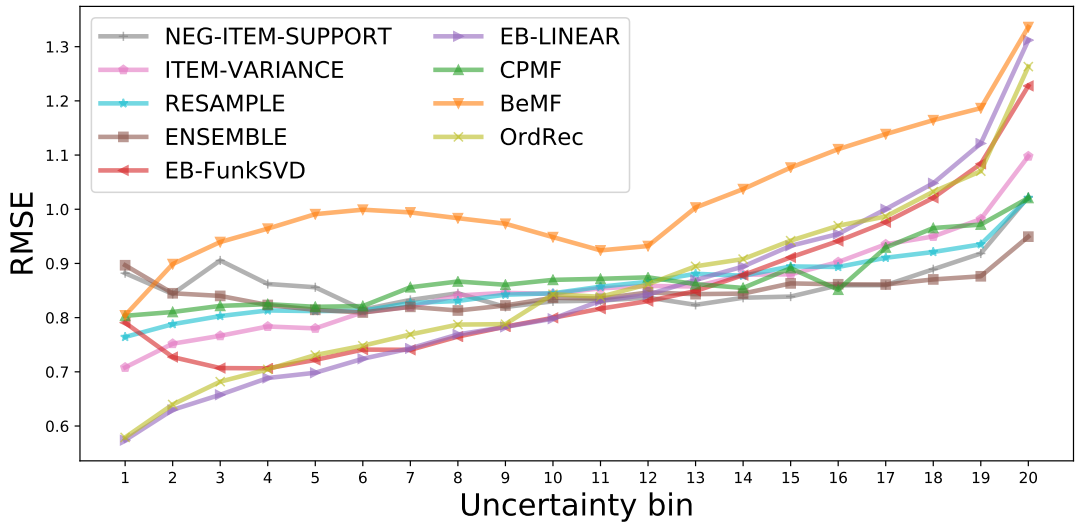
It is worth mentioning that, as explained in Section 3.2, CPMF, BeMF and OrdRec do not directly optimise RMSE, while the baseline model and the models within the ensemble do. In light of this, CPMF shows good performance for both datasets, while ENSEMBLE is able to outperform the baseline in both cases (keeping in mind that low values are better). OrdRec has a slightly worse performance in the MovieLens dataset, while being closer to the other models in the Netflix dataset. BeMF has the worst performance in both datasets.

To assess the quality of the uncertainty estimates from a prediction error point of

view, we employ the methods described in Section 3.3.2. The RMSE-uncertainty curves are given in Figure 3.3. On the whole, we see what we would expect: as uncertainty grows so does error. In both datasets, EB-Linear seems to be the curve that shows this most strongly. In the Netflix dataset, OrdRec and EB-FunkSVD also exhibit this behaviour quite strongly. However, as stated before, these curves are relatively hard to interpret, and for this reason we refrain from drawing strong conclusions from them and turn our attention to the various other evaluation metrics that we presented in Section 3.3.2. The values for these metrics are shown in Tables 3.4 and 3.5.



(a) MovieLens dataset.



(b) Netflix dataset.

Figure 3.3: RMSE-uncertainty curves.

Table 3.4: Rating prediction uncertainty evaluation: MovieLens dataset. The best values (highest) are highlighted in bold.

	Pearson_ρ	Spearman_ρ	$\delta RMSE$	UPI	EUC
NEG-ITEM-SUPPORT	0.0218	0.0391	0.5064	0.1370	0.5353
ITEM-VARIANCE	0.1266	0.0958	0.3034	0.6029	0.5674
RESAMPLE	0.1187	0.1196	0.3243	0.5205	0.5788
ENSEMBLE	0.0327	0.0215	-0.0865	0.0875	0.5051
EB-FunkSVD	0.1788	0.2008	0.0310	0.5500	0.6245
EB-Linear	**0.3463**	**0.2928**	**0.9675**	**1.6851**	**0.6982**
CPMF	0.0449	0.0863	0.3009	0.2444	0.5632
BeMF	0.1114	0.1189	0.3022	0.3204	0.5859
OrdRec	0.2260	0.2526	0.5552	0.6924	0.6731

Table 3.5: Rating prediction uncertainty evaluation: Netflix dataset. The best values (highest) are highlighted in bold.

	Pearson_ρ	Spearman_ρ	$\delta RMSE$	UPI	EUC
NEG-ITEM-SUPPORT	0.0111	0.0277	0.1408	0.0233	0.5170
ITEM-VARIANCE	0.1340	0.1089	0.3897	0.5075	0.5742
RESAMPLE	0.0872	0.0753	0.2567	0.3632	0.5483
ENSEMBLE	0.0405	0.0262	0.0527	0.1761	0.5152
EB-FunkSVD	0.2161	0.1884	0.4370	0.8133	0.6218
EB-Linear	**0.2832**	0.2398	**0.7382**	**1.0839**	0.6587
CPMF	0.0752	0.0684	0.2174	0.2362	0.5424
BeMF	0.1585	0.1755	0.5319	0.3726	0.5949
OrdRec	0.2788	**0.2674**	0.6843	0.7673	**0.6669**

The best overall results are obtained by EB-Linear and OrdRec, while the worst results come from ENSEMBLE. In general, the different metrics in the table seem quite correlated. That is, the performances of the models have nearly the same ordering across all metrics and both datasets. We find these results to be reassuring, since all the metrics are designed to measure the strength of the relationship between uncertainty and rating prediction error. On this, a few observations can be made. EB-Linear's uncertainty seems to correlate linearly with error since their Pearson correlation is stronger, while OrdRec's might correlate non-linearly given the higher Spearman correlation. Moreover, EB-Linear performed strongest according to UPI, while the results for other metrics, such as EUC, are closer, with even a slight advantage for OrdRec in the Netflix dataset.

3.5.3 Top-n recommendation

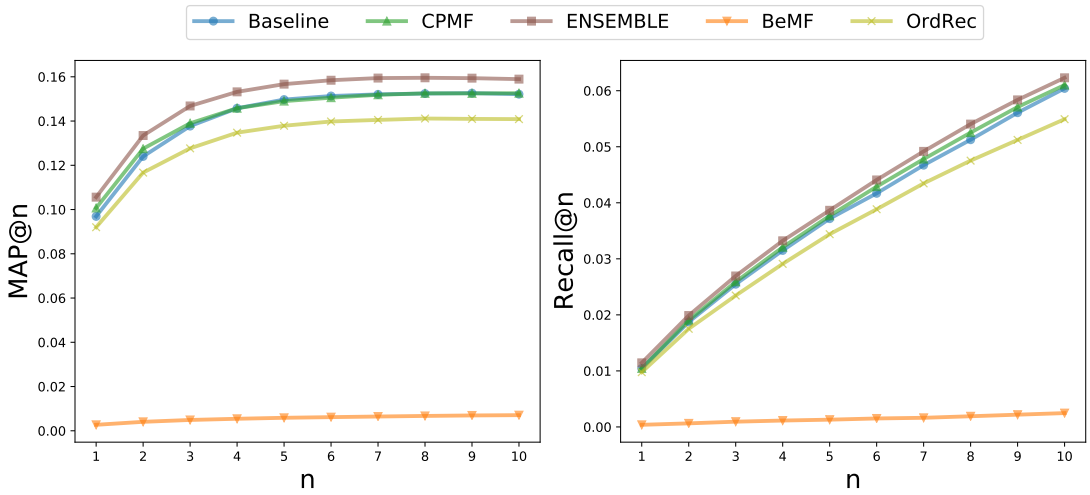
The results in the previous section evaluate the measures of uncertainty through the prism of prediction error. But, as we noted in Section 2.4, we care more about the quality of the top-n recommendations than the prediction error. Hence, using the ideas from Section 3.3.3, we here compare the uncertainty estimates in a variety of top-n recommendation scenarios.

3.5.3.1 Rating-based ranking

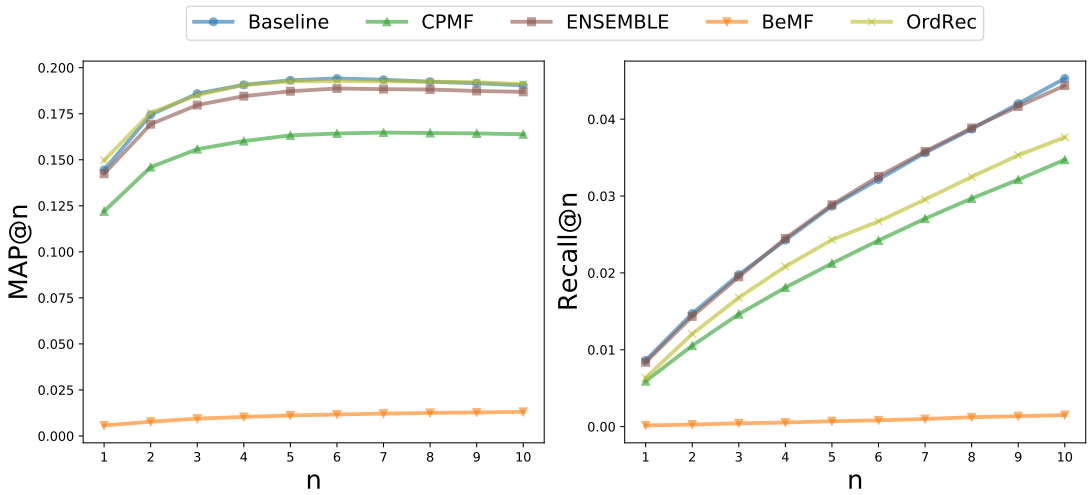
In Rating-Based Ranking, the top-n are the n candidate items with the highest predicted ratings. We begin our evaluation by comparing the accuracy of the different models, leaving uncertainty aside. Figure 3.4 shows the MAP@n and Recall@n, averaged over all test users, for different values for $n \in 1, 2, \ldots, 10$.

The most evident result in the Figure is the very low performance of BeMF in these metrics. This is to be expected. It happens because BeMF is only able to predict rating scale values, that is, $\hat{r}_{ui} \in S$; unlike the other models, it cannot predict a rating that lies between two values on the scale. Therefore, many candidates will receive the same predicted rating – they 'tie'. The creation of a top-n from such an ordering involves a lot of random tie-breaking. This is presumably why BeMF is not evaluated for RaBR in [OLCGPB21]. Apart from BeMF, the remaining models perform similarly, which is reassuring. But there are exceptions: OrdRec has worse MAP for the MovieLens dataset; CPMF is notably worse for the Netflix dataset; and both OrdRec and CPMF have poor recall on the Netflix dataset.

Turning our attention now to the evaluation of the uncertainty estimates, in the RaBR scenario our evaluation tools are URI and UAC. The results are shown in Table 3.6. Remember that higher (positive) URI values denote better performance, while the opposite is the case for UAC. We first notice that the two metrics mostly agree. For both of them, BeMF showed better performance in the MovieLens dataset (although we know that its results are affected by the random choices in the cases of tied ratings), while CPMF was best in the Netflix dataset. On the negative side, RESAMPLE and ENSEMBLE showed poor performance, together with EB-FunkSVD and EB-Linear, which was one of the best performers on the rating prediction task. Therefore, we see a clear disconnect between the models' ability to estimate uncertainty on the rating prediction and top-n recommendation tasks.



(a) MovieLens dataset.



(b) Netflix dataset.

Figure 3.4: RaBR: MAP@n and Recall@n for $n \in 1, 2, \ldots, 10$. (Results for NEG-ITEM-SUPPORT, ITEM-VARIANCE and RESAMPLE will be the same as the Baseline.)

3.5.3.2 Uncertainty-based filtering

In Uncertainty-Based Filtering (UBF), we prevent uncertain candidates (ones whose uncertainty estimate exceeds τ) from being included in the top-n. In the results that we report here, we progressively exclude bigger fractions of the most uncertain candidates, reducing the overall uncertainty of the recommendation lists. More specifically, for each uncertainty estimate, we obtain a distribution using 100,000 randomly-selected user-item pairs and we obtain the 20%, 40%, 60% and 80% percentiles of these distributions as cut-offs for uncertainty. We have chosen this percentile approach due to the different ranges of values that the

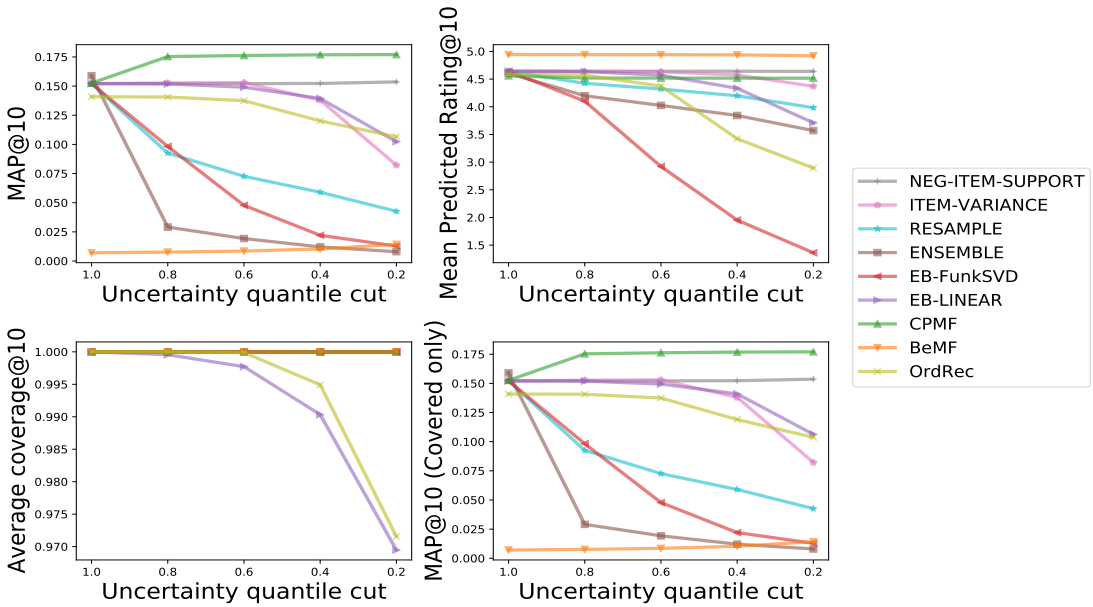
Table 3.6: Rating-Based Ranking uncertainty evaluation: URI and UAC on MovieLens and Netflix datasets. The best values (highest on URI and lowest on UAC) are highlighted in bold.

	MovieLens		Netflix	
Model	URI	UAC	URI	UAC
NEG-ITEM-SUPPORT	0.5070	0.0135	0.6747	-0.1292
ITEM-VARIANCE	-0.0590	0.0316	-0.1561	0.2121
RESAMPLE	-0.2371	0.1138	-0.3554	0.1470
ENSEMBLE	-0.3399	0.2125	-0.4491	0.4141
FunkSVD-CV	-0.0867	-0.0218	-0.2825	0.1733
Bias-CV	-0.1548	-0.0489	-0.2983	0.0037
CPMF	0.5196	-0.0242	**0.6944**	**-0.2905**
BeMF	**0.9527**	**-0.1619**	0.6043	-0.0492
OrdRec	0.2833	-0.0283	0.3808	-0.2143

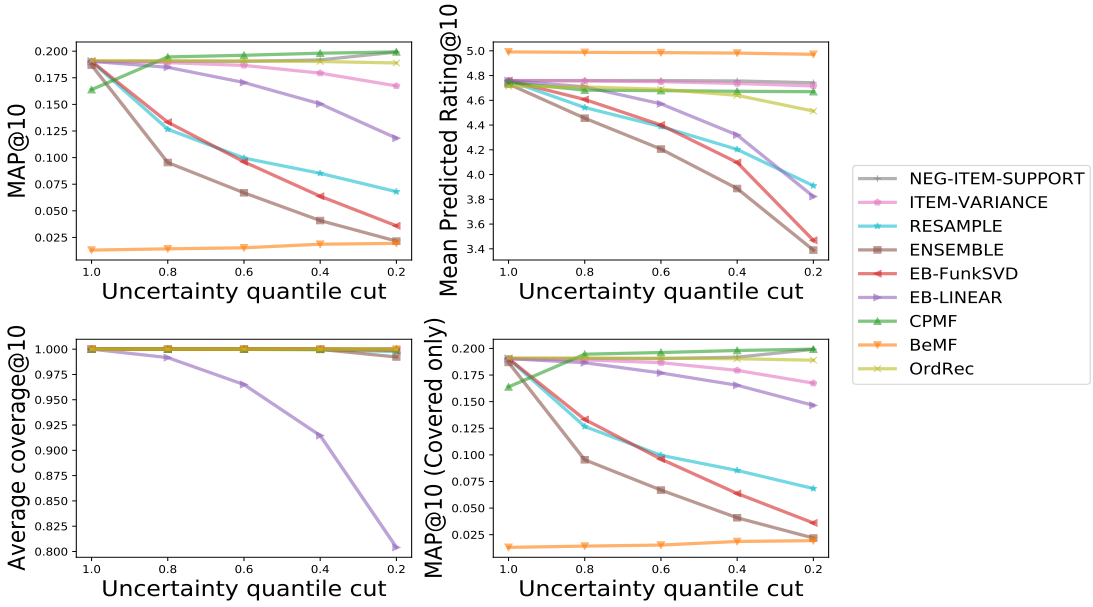
uncertainty estimates have. Figure 3.5 shows the MAP@10 and Mean Predicted Rating@10 of the recommendation lists for these different cut-offs. The same Figure also shows the recommendation coverage with each cut applied and a MAP@10 value that considers only users to whom a full 10 recommendations can be given.

The only uncertainty estimates whose MAP increases as uncertain recommendations are excluded are CPMF, BeMF and, in the case of the Netflix dataset only, NEG-ITEM-SUPPORT. Many of the other uncertainty estimates harm precision as uncertain recommendations are excluded. These results are similar to those we observed with URI in the previous section. An exception is OrdRec, which had high URI but did not perform well here. We also see that the decrease in MAP is, in general, related to the decrease in average relevance as stricter thresholds are applied. That is, the five models with the largest average relevance decrease (ENSEMBLE, EB-FunkSVD, RESAMPLE, ITEM-VARIANCE and OrdRec), are the same ones with the largest drops in MAP. On the other hand, with some uncertainty metrics, such as CPMF and BeMF, it is possible to apply cuts which reduce average relevance only slightly and increase MAP.

Empirically, UBF does not harm coverage. For most of the uncertainty estimates, we can still make 10 recommendations to each user, even as filtering is made stricter. The exceptions are EB-Linear and OrdRec (and the latter only for the MovieLens dataset), and then, even in the worst case (EB-Linear at the 80% quantile cut-off), the model is still able to provide 80% of the requested recommendations. Therefore, the MAP@10 values for users whose coverage is



(a) MovieLens dataset.



(b) Netflix dataset.

Figure 3.5: UBF: MAP, Mean Predicted Rating, Coverage and the MAP for users for whom a full set of recommendations can be made. This is for top-10 recommendations with increasing τ to filter different quantiles.

unharmed are very similar to the original ones.

3.5.3.3 Probability-of-relevance ranking

In Probability-of-Relevance Ranking (PRR), we generate recommendations by ranking items according to the probability that their rating is greater than or equal to the relevance threshold, ω, which for these datasets is 4. As we discussed in Section 3.1.2.2, this idea was proposed for BeMF. But it applies directly also to CPMF and OrdRec. The remaining (non-probabilistic) models, appear to be unsuitable to PRR. Nevertheless, in order not to exclude all of them from the comparisons, we can use some heuristic extensions. The ENSEMBLE trains several models, which can be thought of as an N-dimensional sample of the theoretical population of the rating estimator under all the possible random initialisations. This setup is reminiscent of the central limit theorem. Then, assuming normality, we can use:

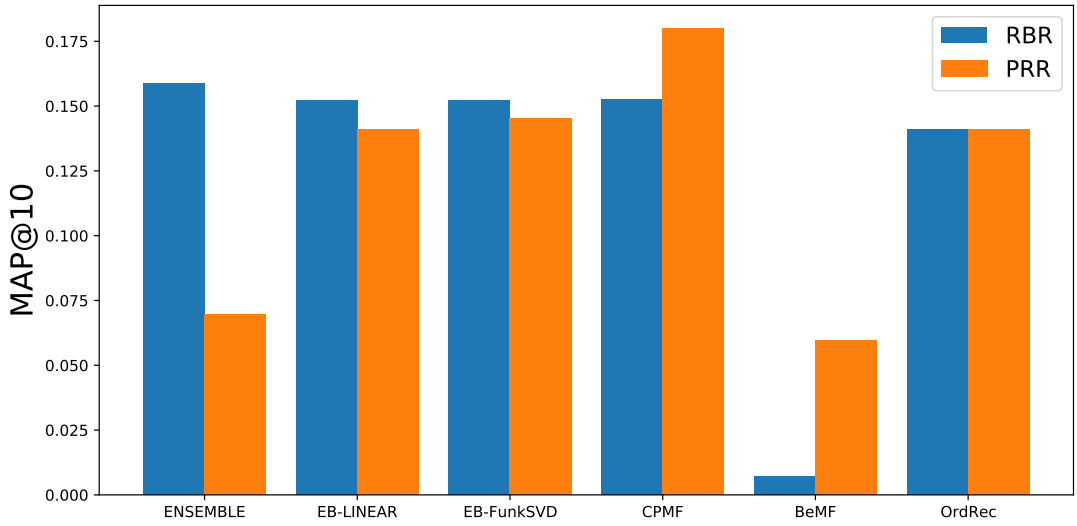
$$\mathbb{P}(r_{ui} \geq \omega) = \mathbb{P}\left(\mathcal{N}\left(\frac{1}{N}\sum_{k=1}^{N} F^{(k)}(u,i), \frac{G_{ui}}{\sqrt{N}}\right) \geq \omega\right) \tag{3.31}$$

Furthermore, the error-based models predict rating prediction errors, that is $G_{ui} = E[|\hat{r}_{ui} - r_{ui}|]$. Therefore, G_{ui} is an estimate of the ratings' standard deviation. Now, assuming r_{ui} to follow a Normal distribution with average $F(u,i)$ and standard deviation G_{ui}, it becomes possible to estimate $\mathbb{P}(r_{ui} \geq \omega)$ as:

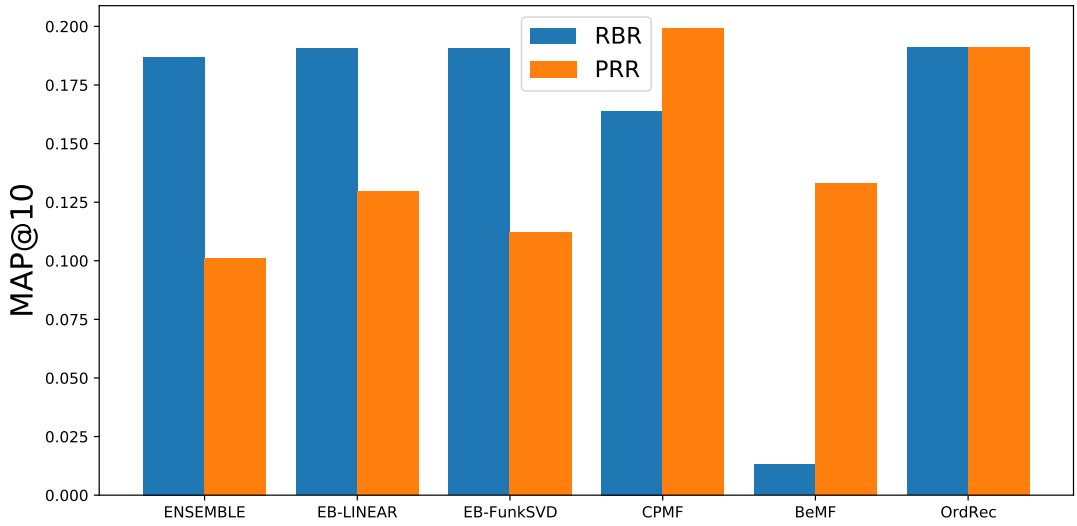
$$\mathbb{P}(r_{ui} \geq \omega) = \mathbb{P}\left(\mathcal{N}\left(F(u,i), G_{ui}\right) \geq \omega\right) \tag{3.32}$$

With such extensions, we can now compare ENSEMBLE, EB-FunkSVD, EB-Linear, CPMF, BeMF and OrdRec for the task of PRR. As discussed in Section 3.3.3.3, we evaluate whether PRR, which uses the uncertainty estimates for ranking, produces more accurate recommendation lists than RaBR, which ignores the uncertainty estimates. This comparison is shown using MAP@10 in Figure 3.6.

In the figure, we see that only CPMF and BeMF have higher MAP@10 when their recommendations are selected using PRR compared with selection by RaBR. OrdRec's MAP@10 is the same for both. For all the other uncertainty estimates, MAP@10 is harmed. This may be because the way that we extended PRR (from BeMF) to these other models is not well-motivated enough, but it also reflects the poor performance of these models.



(a) MovieLens dataset.



(b) Netflix dataset.

Figure 3.6: PRR versus RaBR: MAP@10.

Finally, as discussed in Section 3.1.2.2, the quantity $1 - \mathbb{P}(r_{ui} \geq \omega)$ can be considered as an uncertainty estimate. To evaluate the usefulness of these estimates, we employ UAC, remembering that URI is not applicable in this case due to the correlation between the ranking probabilities and the uncertainties. The results are shown in Table 3.7. The best performances are achieved by BeMF in the MovieLens dataset, and OrdRec in the Netflix dataset. For CPMF, this uncertainty estimate is uncorrelated with MAP. Unfortunately, for ENSEMBLE, EB-FunkSVD and EB-Linear, the uncertainties are positively correlated with MAP.

Table 3.7: PRR uncertainty: UAC on MovieLens and Netflix datasets. The best values (lowest) are highlighted in bold.

Model	MovieLens	Netflix
ENSEMBLE	0.2979	0.2787
EB-FunkSVD	-0.0193	0.2296
EB-Linear	0.0075	0.1180
CPMF	-0.0334	-0.0140
BeMF	**-0.2077**	-0.1509
OrdRec	-0.0237	**-0.2048**

This concludes our results. To summarise, we have shown that there is no model that performs better than all the others across all the metrics that we have explored. Nevertheless, we have shown CPMF to be the best model for both use cases (i.e. for Uncertainty-Based Filtering and Probability-of-Relevance Ranking). On the other hand, our new EB-Linear model had the best uncertainty estimates for most of our rating prediction results, while performing very poorly on the top-n. This highlights a disconnect between the rating prediction task and the top-N recommendation task in the context of uncertainty estimation. Moreover, we saw that the uncertainty estimates behave very differently from each other. This, together with the variations in the performance of the uncertainty estimates across different metrics, raises a major concern about the credibility of some of the uncertainty estimates. Finally, given that top-n recommendations is more important than rating prediction, we believe CPMF to be the most promising of all the models.

3.6 Discussion

So far, we described and empirically evaluated several ways of estimating rating prediction uncertainty. The results show the performance of each uncertainty estimate according to different evaluation criteria. Although we have already raised some general concerns, the results allows us to conduct a further discussion regarding each uncertainty estimate. Furthermore, there are aspects that may be particular to each estimate that may not be explicitly demonstrated by the results. This section discusses these issues and critiques each of the uncertainty estimates.

NEG-ITEM-SUPPORT**:** Characterising the uncertainty of a rating prediction by the number of ratings of the target item has a solid foundation. Its high URI confirms that, when recommended, well-supported items tend to be

more relevant. However, its strong performance on this metric might be attributable to the various biases known to exist in the datasets used for offline RS evaluation [CSE18]. Furthermore, as we have already noted, one concern about NEG-ITEM-SUPPORT is that this estimate of uncertainty is not personalised, which does not seem reasonable; we would expect uncertainty to depend on the relationship between an item and the user profile. Another unreasonable property is its model-independence; we would expect uncertainty to depend in part on the quality of the rating prediction model. Another potential issue is that, if used for UBF, NEG-ITEM-SUPPORT will reduce the item candidates to a subset of the most popular ones, greatly decreasing the system's ability to recommend novel, yet relevant, items.

ITEM-VARIANCE**:** Disagreement between users would certainly seem to be a source of uncertainty. In line with this, the positive results achieved by ITEM-VARIANCE in the rating prediction uncertainty evaluation show that the predicted ratings for items about which users disagree tend to be less accurate. On the other hand, we did not find ITEM-VARIANCE to be useful in the top-n recommendation case. In addition to this, and similarly to NEG-ITEM-SUPPORT, ITEM-VARIANCE is also model-independent and not a personalised metric.

RESAMPLE**:** We found that the variation in predictions that ensues from perturbations during model training does correlate with the error in the final predictions, as evidenced by the positive values achieved by RESAMPLE in the rating prediction uncertainty evaluation metrics. On the other hand, this method obtained negative URI, meaning that, in the task of top-n recommendation, quantifying uncertainty by this method is not only ineffective but actually harmful. One possible explanation for this is that recommended items are the ones with the highest ratings, which means they fall far from the average rating case and are therefore more likely to get noisy predictions.

ENSEMBLE**:** We found ENSEMBLE to have similar, or even worse, results than RESAMPLE in regards to uncertainty evaluation. In fact, given the similar motivation of both, we can conclude that multi-modelling strategies are not useful for uncertainty quantification in the case of top-n recommendation. Nevertheless, we highlight that, in the MovieLens dataset, ENSEMBLE had the best MAP and Recall in the RaBR task, although the improvement relative to the baseline is small.

EB-FunkSVD**:** We expect uncertainty to be a quantity related to rating prediction error. The error-based methods build on this idea directly: they use regression to predict error, and treat these predicted errors as estimates of uncertainty. Unsurprisingly then, EB-FunkSVD achieves good results in the rating prediction uncertainty evaluation. Unfortunately, when it comes to top-n recommendations, it performs poorly. This may be because EB-FunkSVD minimises the average absolute error, but the recommended items are those with the highest ratings, which are outliers relative to the average, and therefore, might not receive good uncertainty predictions.

EB-Linear**:** Our critique of EB-FunkSVD also applies to EB-Linear. In this case, the results are even more evident: the rating prediction uncertainty evaluation results are even better than those of EB-FunkSVD, but its URI results are even worse. Therefore, we conclude that the simple linear model can accurately recognize the user-item interactions where rating prediction imprecision is most likely to occur, but cannot recognize which recommendations are more or less likely to be correct. To solve this issue, one possibility to be explored in future work is to restrict the error estimation learning only to those interactions which get predicted with the highest ratings.

CPMF**:** CPMF extends the popular PMF model by incorporating variance parameters. Whereas other ways of measuring uncertainty are largely heuristic in nature, CPMF has a great theoretical foundation. Not only that, but our experiments show CPMF to be a successful approach, being the only model able to benefit from the uncertainty-aware recommendation methods (UBF and PRR). Nevertheless, its performance in the, admittedly less important, rating prediction uncertainty evaluation was not impressive. Of course, this may not be a weakness of CPMF; it may, instead, raise a flag about the disconnect between prediction error experiments and top-n recommendation experiments. Of the two, a good performance at top-n recommendation is more important than performance in rating prediction.

BeMF**:** In its original formulation, this model has several particularities that contrast with the others herein. For instance, the rating prediction function of BeMF differs from those of OrdRec and CPMF by focusing on the mode of the rating distributions instead of their average and dispersion. We are grateful to the authors of this method, since their work greatly helped our definitions of UBF and PRR. Nevertheless, the results achieved by their model have been largely underwhelming; although the uncertainty

evaluation metrics have been positive in both the rating prediction and the recommendation case, the poor MAP obtained in both RaBR and PRR makes it really hard to justify its use compared, for example, with CPMF, especially because BeMF has much higher complexity.

OrdRec**:** OrdRec offers a collaborative filtering algorithm that is suitable to ordinal feedback data. Although this was its main goal, its authors recognized that its multinomial formulation allows the model to also provide uncertainty estimates. In Section 3.2.5, we raised a concern regarding OrdRec's uncertainty estimates at the limits of the rating scale, which may cause issues. In particular, recommended items are usually those whose predicted ratings fall close to the rating scale upper boundary, and therefore, these items will often receive very low uncertainty estimates, which might not only not reflect reality, but also harms some of the evaluation metrics employed herein (e.g. URI). Furthermore, we have empirically shown that RaBR and PRR produce similar results with OrdRec. In light of this, we believe that these uncertainty estimates add small value when compared to other models.

3.7 Conclusions and Limitations

Uncertainty in ERS is a very important, and yet quite under-explored, topic. In this chapter, we have shown how it is possible to quantify the uncertainty in rating predictions and top-n recommendations using several different methods. Our main findings are:

- We found that all the explored methods are successful at estimating uncertainty in rating predictions, at least to some degree (see Tables 3.4 and 3.5). Among the methods, we found EB-Linear to be the most precise.
- Their success can motivate the use of uncertainty estimates in a variety of tasks in the RS field, including active learning, co-training and reinforcement learning.
- We have also shown that it is possible to improve top-n recommendation accuracy with what methods that we collective refer to as uncertainty-aware ranking. That is, methods that incorporate uncertainty estimates in the selection of a top-n. We showed CPMF to be the best contender for both Uncertainty-Based Filtering (UBF) and Probability-of-Relevance Ranking (PRR).

- In addition, we found that some of the other models, e.g. EB-FunkSVD and EB-Linear, have a poor performance on estimating the uncertainty of top-n recommendations, which makes it hard to defend their use in practical systems, despite their success on rating prediction uncertainty estimation.

Even though the collection of uncertainty estimators explored in this work is the largest yet and the empirical analysis is substantial, we recognize the empirical analysis has some limitations. In Chapter 6, we discuss ideas for future work, some of which follow from the following limitations.

- First, both datasets that we have used come from the same domain (movies). Movie recommendation has historically been one of the most studied applications of recommender systems, but there are many others, such as shopping, social media, etc. While we expect the behaviour of the uncertainty estimators to generalize to other domains, there is clear value in exploring how different rating patterns will affect the uncertainty estimates.
- Second, our empirical analysis uses only a single train/test split. This is a consequence of our decision to opt for chronological train/test splits. We chose, in other words, to preserve the temporal nature of the data, which is intrinsic to recommender systems. This decision does, however, mean that our results lack the kind of statistical significance analysis that could be obtained from using multiple train/test splits. While we expect our results to be robust, due to the large size of our datasets, there is clear value to exploring this further.
- Finally, in our empirical analysis, for those uncertainty estimation methods that need a separate rating predictor, we used FunkSVD (Eq. 2.1) with mean-squared rating prediction error as its loss function (Eq. 2.7). There may be value in exploring other loss functions (e.g. ranking-based losses). We believe our choice was appropriate because it gave a reasonably fair comparison across the systems; the exploration of uncertainty does not rely on finding recommenders with the very highest accuracy; and, in any case, ranking-based loss becomes more relevant when dealing with implicit feedback data, which is the main focus of the next two chapters.

Chapter 4

Prediction Uncertainty in Implicit Feedback-Based Recommender Systems

Implicit feedback-based Recommender Systems (IRSs) have reshaped the field by inferring user preferences from their item interaction behaviour. In contrast to explicit feedback, which involves direct user input such as ratings, implicit feedback focuses on actions such as clicks, downloads and purchases. This approach is valuable in scenarios where explicit feedback is limited. IRSs use these interactions to infer user preferences, offering seamless and unobtrusive insights. This has led to advancements in machine learning techniques and collaborative filtering algorithms, enhancing personalized recommendations across many domains. In an evolving digital landscape, IRSs continue to play a crucial role in delivering tailored content, boosting user engagement, and enriching the overall user experience.

Because of their current prominence, it is natural for the research on prediction uncertainty[1] in RS to be carried over to the implicit feedback case. The available literature in this case is even more limited. For this reason, in this chapter, we:

- Generalize uncertainty estimation methods from Chapter 3, identifying which are directly applicable to IRSs.
- Generalize uncertainty evaluation metrics from Chapter 3, distinguishing those that remain relevant in the case of IRSs.

[1] As per the previous chapter, here we refer to prediction uncertainty simply as uncertainty throughout.

- Describe the existing methods for uncertainty estimation for IRSs, and propose novel uncertainty estimation methods. In particular, due to their increasing usage in the field, we propose methods for uncertainty estimation in neural network-based IRSs.
- Present an extensive comparison between all the methods surveyed in the chapter and all the novel methods proposed in the chapter.

4.1 Prediction and Recommendation in IRSs

In this chapter, we will explore RS based on implicit feedback data, specifically those based on unary interaction data (e.g. clicking, purchasing, watching, listening) and with no explicit feedback given. Now, in the absence of ratings, we assume that our data consists of a set R of user-item interaction pairs. That is, if $\langle u, i \rangle \in R$, then user $u \in U$ has interacted with item $i \in I$.

4.1.1 Relevance prediction

Essentially, IRSs are designed to retrieve, from a large catalog, those items that are most relevant to each user. To achieve its goal, the system must learn a scoring function $F : U \times I \to \mathbb{R}$ to predict the user-item interaction relevance. Note how this differs from Explicit feedback-based Recommender Systems (ERSs): an ERS make rating predictions; for an IRS rating prediction no longer applies; scoring is based on predicted relevance. In this chapter, for relevance predictions, we confine our attention to model-based Collaborative Filtering (CF) algorithms, in particular, traditional Matrix Factorization (MF) and Neural Network (NN) models.

In addition to relevance, we want to estimate the uncertainty associated with each relevance prediction. Again, an uncertainty estimator is a function $G : U \times I \to \mathbb{R}$, that predicts real-valued quantities associated with each relevance prediction.

4.1.2 Top-n recommendation

The task of top-n recommendation in IRSs is analogous to the explicit case, but using predicted relevance instead of predicted ratings. Hence, the conventional way to retrieve a set of recommendations Z_u^n for user $u \in U$ with candidate items $C_u \subseteq I$ in an IRS is to select the top-n candidates with highest predicted relevance. We denote this form of ranking Relevance-Based Ranking (ReBR).

But, similarly to the previous chapter, there are uncertainty-aware alternative to ReBR. The uncertainty estimates may again assist in ranking and selection of a top-n. In particular, Uncertainty-Based Filtering (UBF), introduced in Section 3.1.2.1, is applicable to IRSs with no further adaptation. On the other hand, Probability-of-Relevance Ranking (PRR) is not applicable because scoring in IRSs is by default already based on relevance.

4.2 Estimating Uncertainty

Recommendation uncertainty has several causes, including sparsity of data, modeling choices, and stochastic learning algorithms. For this reason, methods for uncertainty estimation in the field are very diverse. Therefore, similar to Section 3.2, it helps to organize the uncertainty estimation methods according to their uncertainty source.

Information-based: The amount of available data continues to be a fundamental source of uncertainty in implicit feedback data.

Stability-based: As previously, the stability of the learning algorithms can be used to estimate uncertainty. In particular, stability becomes a serious consideration when using neural-based recommenders, due to the stochastic mechanisms involved in their learning [CAH22, SD21].

Distribution-based: Interaction relevance scores may be assumed to follow a statistical distribution, whose dispersion is an uncertainty estimate. In this case, uncertainty estimation is tackled as a Machine Learning problem, in which the uncertainty estimator is *learned* just like the relevance estimator. Due to particularities of learning uncertainty estimators in the case of IRS, we exclude them from this chapter. We will these particularities in depth and this class of models in Chapter 5.

We note that, in Chapter 3, we classified the uncertainty estimators into five groups, while above we have just three of them. This is because the Error-based and Multinomial-based uncertainty estimators are not defined for IRSs: since there are no ratings, there can be no rating prediction errors (required for Error-based estimators) and discrete rating scales (required for Multinomial-based estimators).

We now present information-based and stability-based uncertainty estimation methods for IRS. In each case, we make clear which recommender algorithms can

be used with each uncertainty estimation method.

4.2.1 Information-based uncertainty

One of the most notable sources of uncertainty is sparsity. For this reason, the amount of available data offers good baseline estimates of recommendation uncertainty [Maz13]. Furthermore, these estimates can be used with any recommendation algorithm. In the past, they have been used for explicit feedback recommenders, but their generalization to IRSs is straightforward. These estimates can be user-centric or item-centric. Hence, following [Maz13], we define the following uncertainty metrics,

$$\text{NEG-USER-SUPPORT} : G_{ui} = -\#R_{u\cdot} \tag{4.1}$$

$$\text{NEG-ITEM-SUPPORT} : G_{ui} = -\#R_{\cdot i} \tag{4.2}$$

where $\#u$ and $\#i$ denote the number of observed interactions for the user and the item, respectively. The clear drawback of these uncertainty estimates is that they are either at user-level or at item-level, that is G_{ui} is defined solely based on the user, or the item, but not on the user-item interaction. Nevertheless, they have the advantage of needing no additional learning and can be easily plugged into any system.

As we noted previously (Section 3.2.1), since top-n recommendation is a user-based task, user-wise uncertainty estimates, such as NEG-USER-SUPPORT, are largely useless. The same reasoning also applies to the case of IRSs, and therefore, we will not explore NEG-USER-SUPPORT any further in this chapter. Naturally, USER-VARIANCE and ITEM-VARIANCE, the other forms of information-based uncertainty from Chapter 3 are not applicable in the absence of explicit ratings.

4.2.2 Stability-based uncertainty

Beyond the uncertainty introduced by the data, every recommender algorithm has its own uncertainty issues. Consider, for example, models that are based on representation learning, such as MF, where vector embeddings are learned as a latent representation for each user and item. For such models, the uncertainty surrounding the learning of such representations will affect the system recommendations. In fact, MF is known to suffer from learning instability [DGBC22, POMT+20].

In the case of explicit feedback, ensembles have been successful at estimating the uncertainty of MF rating predictions [Maz13]. But, explicit feedback MF is only one of many algorithms that can benefit from ensembling. In fact, an ensemble can be used to estimate uncertainty for any model that relies on a stochastic mechanism, such as random parameter initialization or stochastic learning protocols. This is the case for implicit feedback MF and also any neural network model, and in particular the MLP model (Eq. 2.3).[2]

Formally, the principle is to train several models $F^{(k)}$, for $k = 1, \ldots, n$ using a different random initialization each time, and then calculate interaction relevance and uncertainty as follows:

$$F(u,i) = \frac{\sum_{k=1}^{n} F_\theta^{(k)}(u,i)}{n} \tag{4.3}$$

$$G_{ui} = \frac{\sum_{k=1}^{n} \left(F_\theta^{(k)}(u,i) - r_{ui}\right)^2}{n} \tag{4.4}$$

But there are alternatives to ensembling, which we can find in the ML literature and apply to the case of IRSs. We will look at two: Bayesian Neural Networks and MCDropout.

Bayesian Neural Networks (BNN) are a major tool tailored to uncertainty quantification in neural models. BNNs differ from their deterministic counterpart by treating the parameters as random variables [GF20], which are assumed to follow some prior distribution $p(\theta)$. Given some training data $\mathcal{D}$, the posterior weight distribution, according to Bayes rule, is as follows,

$$p(\theta|\mathcal{D}) = \frac{p(\theta)p(D|\theta)}{p(D)} \tag{4.5}$$

Calculating the posterior directly from Eq. 4.5 is generally not possible, because the data evidence, $p(D)$, is unknown. For this reason, inference methods such as Monte-Carlo Markov Chains (MCMC) [Gey92] and Variational Inference (VI) [Gra11] are applied to approximate the exact posterior. More recently, Bayes By Back-propagation (BBB) has been proposed [BCKW15], a method that allows for the posterior weights distribution to be learned through back-propagation,

[2] An ensemble of neural models is often referred to as a Deep-Ensemble [LPB17].

just as the weights of a non-Bayesian network are learned by conventional back-propagation. Predictions can then be made using the estimated posterior.

More precisely, the output's expected value $\mathbb{E}[F_\theta(u,i)]$ is a point prediction for the interaction relevance F_{ui}, and its variance $Var[F_\theta(u,i)]$ is an estimate of the relevance's uncertainty G_{ui}. In practice, the values are estimated using samples $\theta_k, \ldots, \theta_k$ from the posterior, as follows,

$$F_{ui} = \frac{\sum_{k=1}^{n} F_{\theta_k}(u,i)}{n} \tag{4.6}$$

$$G_{ui} = \frac{\sum_{k=1}^{n} \left(F_{\theta_k}(u,i) - r_{ui}\right)^2}{n} \tag{4.7}$$

Another uncertainty estimation method that is tailored to neural networks is MCDropout [GG16]. The method, which can be thought of as an approximation of a Bayesian network, consists of taking multiple forward passes with dropout enabled at prediction time.[3] Formally, let $F^{(k)}$, for $k = 1, \ldots, n$ denote k predictions calculated with dropout enabled. Then, the final estimates for relevance and uncertainty follow according to Equations 4.3 and 4.4.

Interestingly, MCDropout has been used to estimate the uncertainty of a RS in [ZTS+17]. But, to the best of our knowledge, the usage of Deep-Ensembles and BNNs for uncertainty estimation remain unexplored in the field. In Section 4.4, we fill this gap by comparing all of these methods as uncertainty estimators for IRSs.

4.3 Evaluating Uncertainty Estimates

With several uncertainty estimators defined, we now turn our attention to their evaluation. In the previous chapter, we explored a very wide set of evaluation methods for uncertainty estimate in ERSs (see Section 3.3). Naturally, in the absence of ratings, most of them are no longer useful. Nevertheless, all the evaluation criteria for Rating-Based Ranking (see Section 3.3.3.1) can be translated into Relevance-Based Ranking. Furthermore, some of the expected correlations still hold. In this section, we list the uncertainty evaluation metrics that are applicable to IRSs, highlighting any necessary modifications.

[3] Conventionally, dropout is enabled at training time and combats overfitting. In MC-Dropout, it is enabled at prediction time to sample a space of predictions.

4.3.1 Expected correlations

Three of the patterns that we discussed in the previous chapter and that we expect to observe with uncertainty estimates can also be measured in the case of IRS: the negative correlations between uncertainty and user profile size; the negative correlations between uncertainty and item support; and the negative correlation between recommendation uncertainty and accuracy. These expected correlations follow from the same reasoning and can be measured exactly as for ERS.

4.3.2 Top-n recommendation

Because we no longer consider explicit ratings, all the uncertainty evaluation tailored to rating prediction (Section 3.3.2) do no apply, and therefore we jump straight to the top-n recommendation case.

4.3.2.1 Relevance-based ranking

To begin with, we generalize Uncertainty Recommendation Improvement (URI) to make it suitable to IRSs. Its motivation remains the same: recommendations that have lower uncertainty are expected to be more precise. Therefore, URI compares the uncertainty of each item against the average uncertainty in the recommendation list. The only adaptation we must make to URI is to get rid of the relevance threshold ω (from Eq. 3.28), because all the observed interactions are assumed to be equally relevant. Therefore, the URI for IRSs is defined as:

$$C_u^{URI} = \sum_{i \in Z_u^n | i \in R_{u\cdot}} (\overline{\rho}_u - G_{ui}) \tag{4.8}$$

where $\overline{\rho}_u$ is the mean uncertainty for the items in Z_u^n, i.e. the n items that are recommended to user u.

Now, the URI is defined as:

$$URI = \frac{\sum_u C_u^{URI}}{\sigma_{\rho_u} \sum_u \#\{i \in Z_u^n | i \in R_{u\cdot}\}} \tag{4.9}$$

where σ_{ρ_u} is the standard deviation of the uncertainty values for Z_u^n.

4.3.2.2 Uncertainty-based filtering

Of the two types of uncertainty-aware ranking (Uncertainty-Based Filtering and Probability-of-Relevance Ranking), only the former (UBF) extends to IRSs. We recall that UBF is the form of ranking that discards candidate items whose estimated uncertainty is above a threshold τ (see Section 3.1.2.1). Therefore, we can use it again as a form of evaluation. Identically to the description in Section 3.3.3.2, our proposal is to progressively increase the uncertainty threshold for filtering τ while measuring the accuracy after each cutoff. In addition, it is also important to measure the average relevance of the recommended items because, with more candidates discarded, the average relevance of recommendations will drop. We calculate Mean Predicted Relevance as,

$$\text{Mean Predicted Relevance@}n = \frac{1}{\#U} \sum_{u \in U} \frac{1}{n} \sum_{i \in Z_u^n} F_\theta(u, i) \tag{4.10}$$

Finally, we must also measure coverage, to measure the extent to which we are still able to make n recommendations as we discard ever more candidates. The same coverage metric from Eq. 2.27 can be employed.

4.4 Empirical Study

In this section, we aim to compare the uncertainty estimation methods proposed in Section 4.2. To do so, we use the evaluation methods and metrics explored in Section 4.3. For reproducibility, we make all the code used available[4]. Our implementation is based on PyTorch [PGC+17].

4.4.1 Models

Our comparative study in this chapter include the following uncertainty estimators: NEG-ITEM-SUPPORT, MF-ENSEMBLE, MLP-ENSEMBLE, Bayesian-MLP and MCDropout.

These methods rely on a separate, underlying relevance prediction model (F). For this model, we adopt either MF or MLP, as described in Section 2.3.1. To choose the best underlying model in each case, we conduct a comparison between several versions of MF and MLP (see Section 4.5.1). The best performing models will have two uses. First, they will be used as the underlying relevance prediction

[4]https://github.com/vcoscrato/uncertain

Table 4.1: Relevance prediction models that we compare (with equation numbers). These are used for F, but also as baselines in accuracy comparisons.

Notation	Prediction model	Learning objective
MF-MSE	MF (Eq. 2.1)	MSE (Eq. 2.9)
MF-BCE	MF (Eq. 2.1)	BCE (Eq. 2.11)
MF-BPR	MF (Eq. 2.1)	BPR (Eq. 2.13)
MLP-MSE	MLP (Eq. 2.1)	MSE (Eq. 2.9)
MLP-BCE	MLP (Eq. 2.1)	BCE (Eq. 2.11)
MLP-BPR	MLP (Eq. 2.1)	BPR (Eq. 2.13)

Table 4.2: Uncertainty estimation methods that we compare.

Notation	Uncertainty estimator	Relevance model
NIS	NEG-ITEM-SUPPORT (Eq. 4.2)	Best baseline
MF-Ensemble	Ensemble (Eqs. 4.3 – 4.4)	Best MF baseline
MLP-Ensemble	Ensemble (Eqs. 4.3 - 4.4)	Best MLP baseline
BayesianMLP	Bayesian inference (BBB) (Eqs. 4.6 – 4.7)	Best MLP baseline
MCDropout	Monte-Carlo Dropout (Eqs. 4.3 – 4.4)	Best MLP baseline

models; and second, they will serve as a baseline for recommendation accuracy. Our goal is to assess whether the uncertainty estimators will help improve accuracy relative to the baseline or not. Table 4.1 shows the relevance prediction models that we compare.

In Table 4.2, we list all uncertainty estimation methods that we consider, also highlighting the used underlying relevance prediction model. Note that the objective function used to learn the underlying model will be the best one, for each dataset, according to Section 4.5.1.

4.4.2 Datasets

We evaluate our models and uncertainty estimation methods on two datasets: an implicit version of the MovieLens-1M dataset [HK15][5] and one Pinterest dataset [GZBC15]. These are very popular and openly available datasets. Table 4.3 presents some summary statistics.

[5]To make this dataset implicit, we simply treat every given rating as an implicit signal (1), ignoring the numeric rating value.

Table 4.3: Characteristics of the datasets used in our experiments.

Dataset	MovieLens-1M	Pinterest
num. users, $\#U$	6,040	55,187
num. items, $\#I$	3,416	9,643
num. ratings, $\#R$	1 Million	1.5 Million
sparsity	95.15%	99.72%
avg. num. ratings per user	166	27
avg. num. ratings per item	293	156

4.4.3 Methods

4.4.3.1 Data splitting

For both datasets, we use a user-based chronological data splitting method: we chronologically order the interactions of each user and place the last 20% of them in the test set. The remaining 80% of interactions for each user are used for training or validation. More precisely, we order these remaining ratings chronologically; for each user, we use the first 80% of these for training, and the rest for validation.

4.4.3.2 Tuning

MF, MLP and BayesianMLP have hyper-parameters that need to be tuned. We tuned our models using a Bayesian parameter search, assisted by Optuna [ASY+19]. More precisely, we define a parametric search space and train multiple models by sampling parameter settings from the space.[6] For each algorithm, we learned 20 models with different parameter settings and chose the one with the highest MAP@10 (see Eq. 2.20).

We sample hyper-parameters from the following:

- For all models, learning rate is sampled from $[0.0001, 0.01]$.
- For models that involve negative sampling (Point-wise objectives), the number of negative training instances N per positive training instance is sampled from $\{1, 2, ..., 20\}$.
- MF: The L2 penalty factor applied to the user and item factors was taken from $[10^{-5}, 10^{-2}]$.
- MLP: We use the same three-layer MLP as in [HLZ+17]. We also employ

[6] In a Bayesian parameter search, the sampler is initially random and then progressively guided by the performance of the learned models. For details, see [Ngu19].

dropout on the training stage. The dropout rate is tuned in $[0, 0.2]$.

- BayesianMLP: We use a Bayesian MLP with the same architecture as our deterministic MLP. We use the same prior and tune the hyper-parameters related to it across the exact same grid as used in [BCKW15].

For MF-ENSEMBLE and MLP-ENSEMBLE, we set the ensemble size to $n = 5$. We experimented with larger sample sizes ($n = 10$ and $n = 20$), but found they all produced similar results.

The user and item latent embeddings size d to 128 for all models. Setting them to the same size gives a fair comparison. While it has been shown that both MF and MLP can benefit from even higher dimensions [RKZA20], $d = 128$ gives us reasonable computational cost.

Finally, to avoid the need to tune the number of training iterations, we employ early-stopping to end the learning phase when the MAP@10 on the validation set does not improve for three consecutive iterations.

4.5 Results

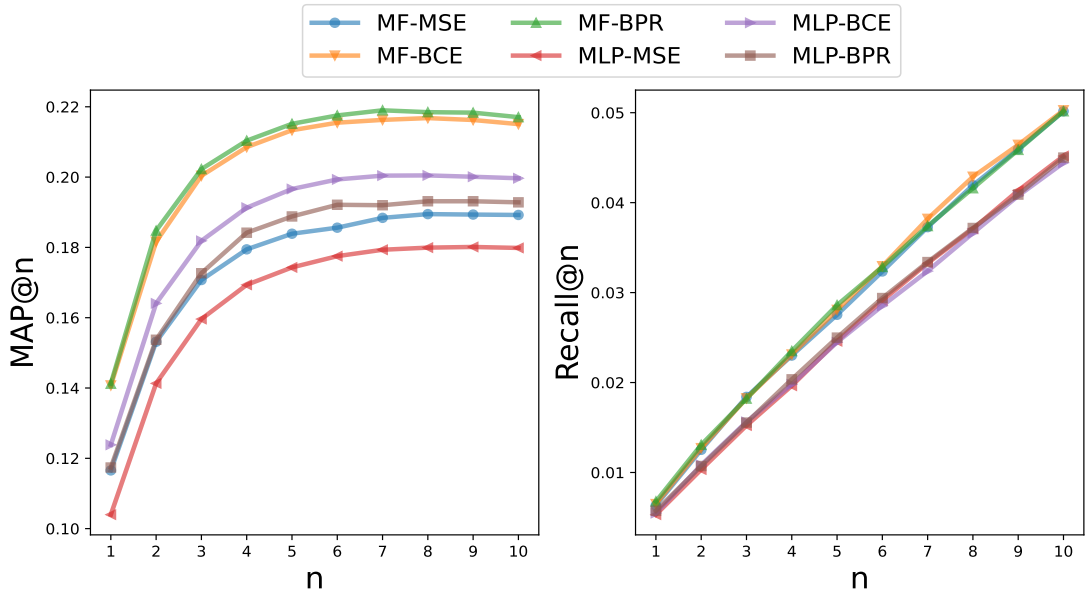
In this section, we present the results[7]. We begin with the comparison between the baselines. From this comparison, we adopt the best performing ones as the underlying relevance prediction models for our uncertainty estimators. Then, we compare the uncertainty estimators using the methods and metrics from Section 4.3.

4.5.1 Baseline models

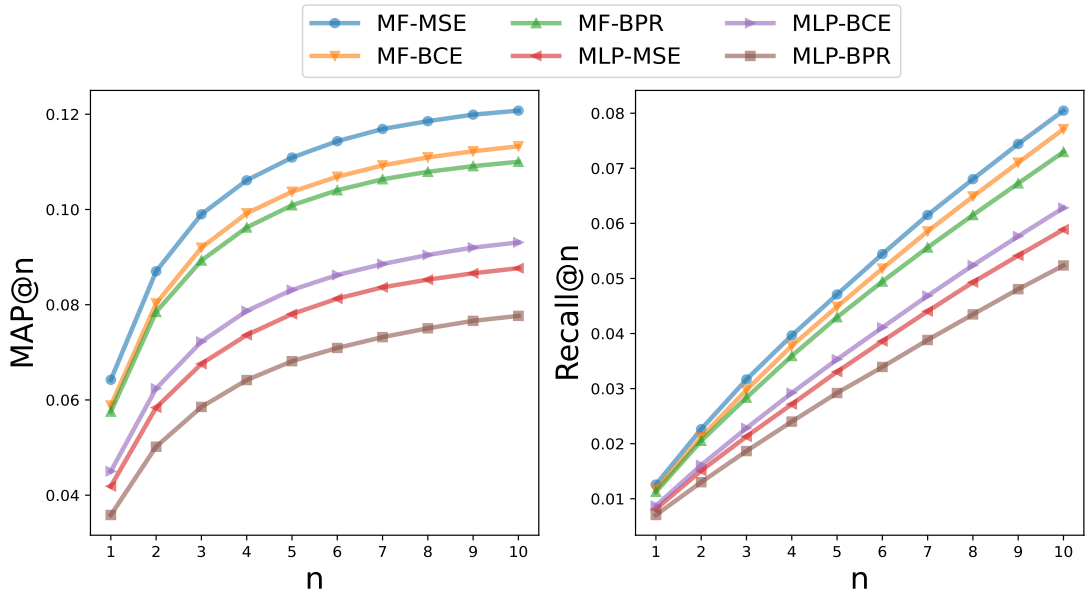
Figure 4.1 illustrates the recommendation accuracy for each baseline. In both datasets, we observe a clear advantage of MF over MLP. This is expected, because the same has been observed in [RKZA20].

For the MovieLens dataset, the MAP and Recall of MF-BPR and MF-BCE are almost indistinguishable, while MF-MSE falls behind. Interestingly, the opposite is observed in the Pinterest dataset: MF is the most accurate method, irrespective of the loss function, according to both MAP and Recall.

[7] We note that the results presented here are not identical to the results that we published in [CB22]. The major difference is that we added a more complex search for optimal baselines (see Table 4.1). As a consequence, there are small variations in the performance of the uncertainty estimators that use the baselines as their underlying models.



(a) MovieLens dataset.



(b) Pinterest dataset.

Figure 4.1: MAP@k and Recall@k, for $k = 1, 2, \ldots, 10$ for all the baseline models on the MovieLens and Pinterest datasets.

Focussing on MLP, MLP-BCE is better than MLP-MSE and MLP-BPR in almost every case. The only exception is the Recall on the MovieLens data, where all MLP methods had almost identical performance.

According to the results obtained, our choice for underlying models are: NIS and MF-Ensemble will use MF-BPR for the MovieLens dataset and MF-MSE for the

Pinterest. MLP-Ensemble, BayesianMLP and MCDropout will use MLP-BCE for both datasets.

4.5.2 Expected correlations

We begin our uncertainty evaluation by analyzing correlations between the uncertainty estimates and: the user profile size ($\#R_{u\cdot}$); and item support ($\#R_{\cdot i}$).

We follow exactly the same procedure from Section 3.5.1. That is, we calculate the correlations from a randomly sampled set of 100,000 user-item pairs. Figure 4.2 illustrates the results. Note that the correlation between NEG-ITEM-SUPPORT and $\#R_{\cdot i}$ is trivially perfect, and therefore ignored.

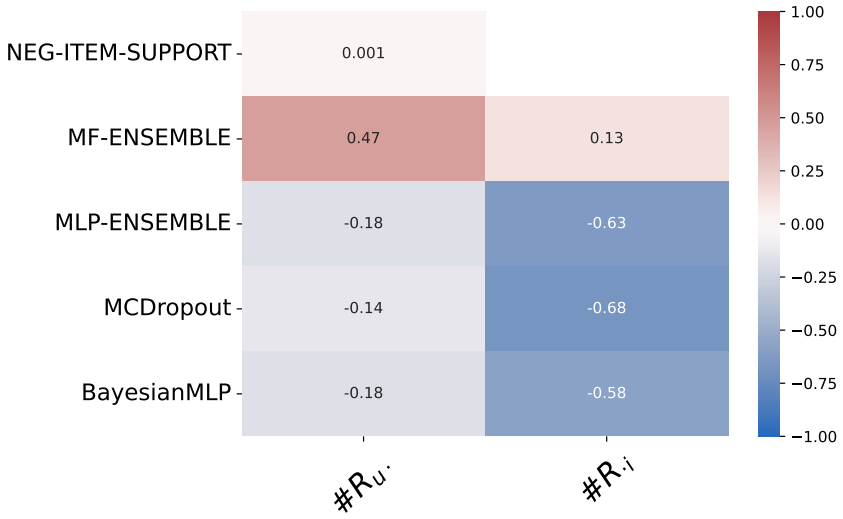
In the MovieLens data, we observe many of the expected correlations: the uncertainties estimated by MLP-ENSEMBLE, MCDropout and BayesianMLP are negatively correlated with both user profile size (weakly) and item support (strongly). The exception to the rule is MF-ENSEMBLE. The uncertainties estimated by the model grow with both $\#R_{u\cdot}$ and $\#R_{\cdot i}$. Interestingly, the same counter-intuitive behaviour was also observed when using a MF-ENSEMBLE for ERSs (see Section 3.5.1). This is also the case for MF-ENSEMBLE in the Pinterest dataset (4.2, b).

Furthermore, in the Pinterest data, we also observe negative correlations between MCDropout and BayesianMLP uncertainties and $\#R_{\cdot i}$ although the magnitude of the correlations are much smaller than for the MovieLens dataset. In addition, these models' uncertainties do not present any noticeable correlations with $\#R_{u\cdot}$. Finally, we do not observe correlations between MLP-Ensemble's uncertainties and either $\#R_{\cdot i}$ or $\#R_{u\cdot}$.

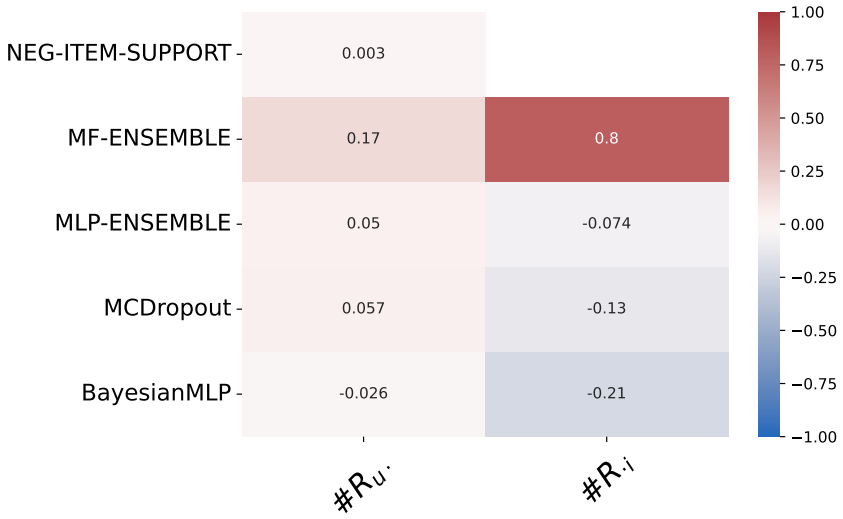
It is reassuring that the three neural models have similar correlation results. After all, MLP-ENSEMBLE and MCDropout are approximations of the BayesianMLP. As for the counter-intuitive results observed with MF-ENSEMBLE in both datasets, these combine with similar results from Section 3.5.1 to reinforce the concerns regarding this as a method for uncertaint estimation.

4.5.3 Top-n recommendation

We now turn our attention to the most important task of a RS: top-n recommendation.



(a) MovieLens dataset.



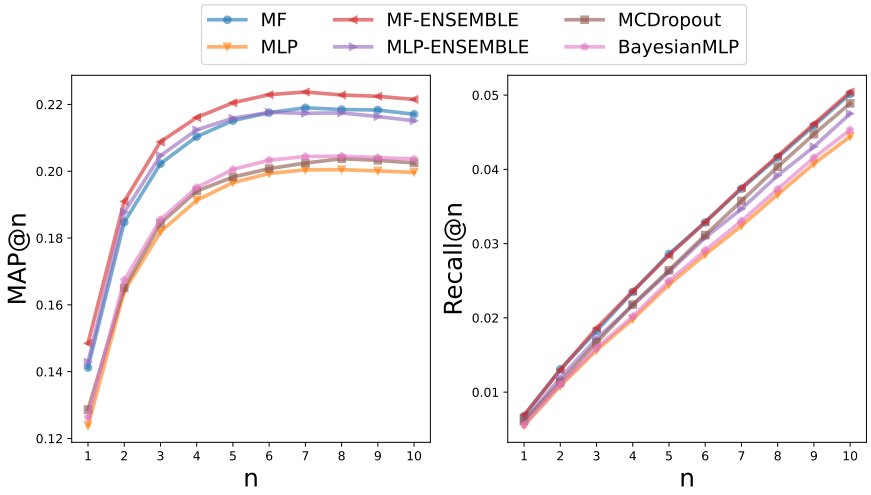
(b) Pinterest dataset.

Figure 4.2: Correlations between uncertainty estimates and dataset statistics. We expect negative correlation with $\#R_{u\cdot}$ and $\#R_{\cdot i}$.

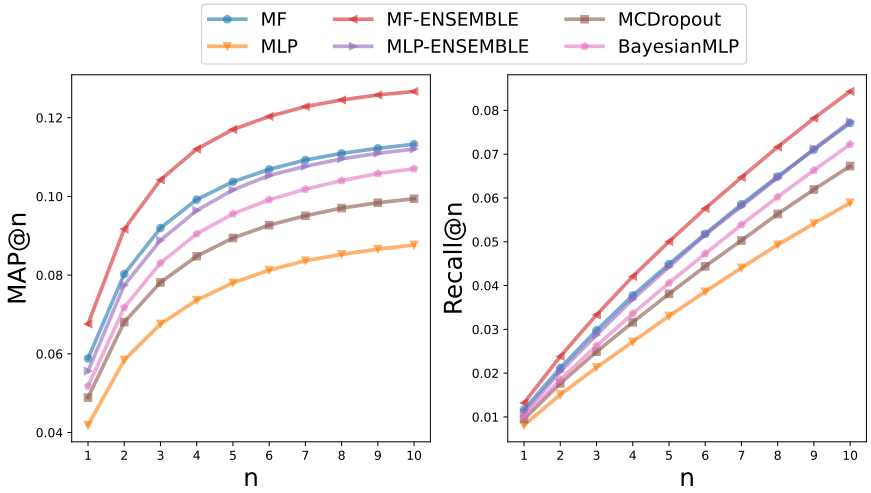
4.5.3.1 Relevance-based ranking

We begin by comparing our models' accuracy for recommendations given using Relevance-Based Ranking (ReBR). To do so, we calculate the MAP@n and Recall@n, averaged over all users, for different recommendation list sizes $n \in 1, 2, \ldots, 10$. Figure 4.3 shows the results.

The foremost result from the graphs is the pronounced advantage of the ensemble-based models, for both datasets and according to both metrics, in comparison



(a) MovieLens dataset.



(b) Pinterest dataset.

Figure 4.3: ReBR: MAP@n and Recall@n for $n \in 1, 2, \ldots, 10$. The MF and MLP correspond to those with highest MAP@10, for each dataset, according to Figure 4.1.

with the baseline models (MF and MLP). Intriguingly, we have repeatedly observed such an advantage (see also Section 3.5.3.1), even though ensembles do not seem great in relation to our uncertainty evaluation, both for ERSs (Section 3.3.3) and IRSs, as we showed in the previous section and as we will confirm later in this chapter.

Moreover, we observe that both MCDropout and BayesianMLP are superior to the corresponding MLP baseline, for both metrics and both datasets. This, together with the success of MLP-ENSEMBLE, gives support to the usage of Bayesian Neural Networks, or their approximations, in RSs, although, we note

Table 4.4: Rating-Based Ranking uncertainty evaluation: URI on MovieLens and Pinterest datasets. The best (highest) values are highlighted in bold.

	MovieLens	Pinterest
NEG-ITEM-SUPPORT	**0.1324**	**0.1306**
MF-ENSEMBLE	-0.0360	-0.1281
MLP-Ensemble	-0.0358	-0.0100
MCDropout	-0.0306	-0.0845
BayesianMLP	0.0238	0.0288

that all MLP-based methods used here are worse than the baseline MF, in all cases.

We now turn our attention to the uncertainty estimates. We aim to verify the validity of the uncertainty estimates provided by all our researched methods on top-n recommendations. To evaluate the uncertainty estimates produced by each method quantitatively, we employ URI. The results are presented in Table 4.4. A positive URI implies that the less uncertain a recommended item is compared to the average uncertainty in the user's recommendations, the more likely it is to be relevant.

Unfortunately, we do not observe a high URI with any of our methods. In fact, we even observe a negative URI (although one of very small magnitude) in many cases. The exception is NEG-ITEM-SUPPORT. In both datasets, NEG-ITEM-SUPPORT was the only model that achieved a considerably positive URI. It means that, the more popular an item is in regards to the average uncertainty in the recommendation list it appears in, the more likely it is to be relevant.

One could argue that these negative results point to a potential problem with the URI metric, rather than with the uncertainty estimators. URI assumes that correct recommendations – that is, the recommendation of relevant items – should be given with lower uncertainty. Although this makes sense, we recall that relevance is inferred through historical interactions, but it is very plausible that at least some unobserved interactions are relevant[8]. In these cases, is it right to evaluate uncertainty estimates through URI? We leave for future exploration the question of whether this is explains the results we have obtained.

[8]In fact, this is the premise of label uncertainty, as introduced in Chapter 1 and explored in detail in Chapter 5.

4.5.3.2 Uncertainty-based filtering

Of more importance than URI is the suitability of the uncertainty estimators to their use cases. In UBF, the uncertainty is used to prevent items from being recommended to users when the user-item pair has high estimated uncertainty. The filtered recommendation lists will have smaller average uncertainty, and are therefore expected to be more precise.

The results reported here follow the exact same procedure used in Section 3.3.3.2. For each uncertainty estimator, we progressively exclude bigger fractions of the most uncertain candidates from the pool, according to cut-offs (τ) that represent the 20%, 40%, 60% and 80% quantiles of their uncertainty distributions.

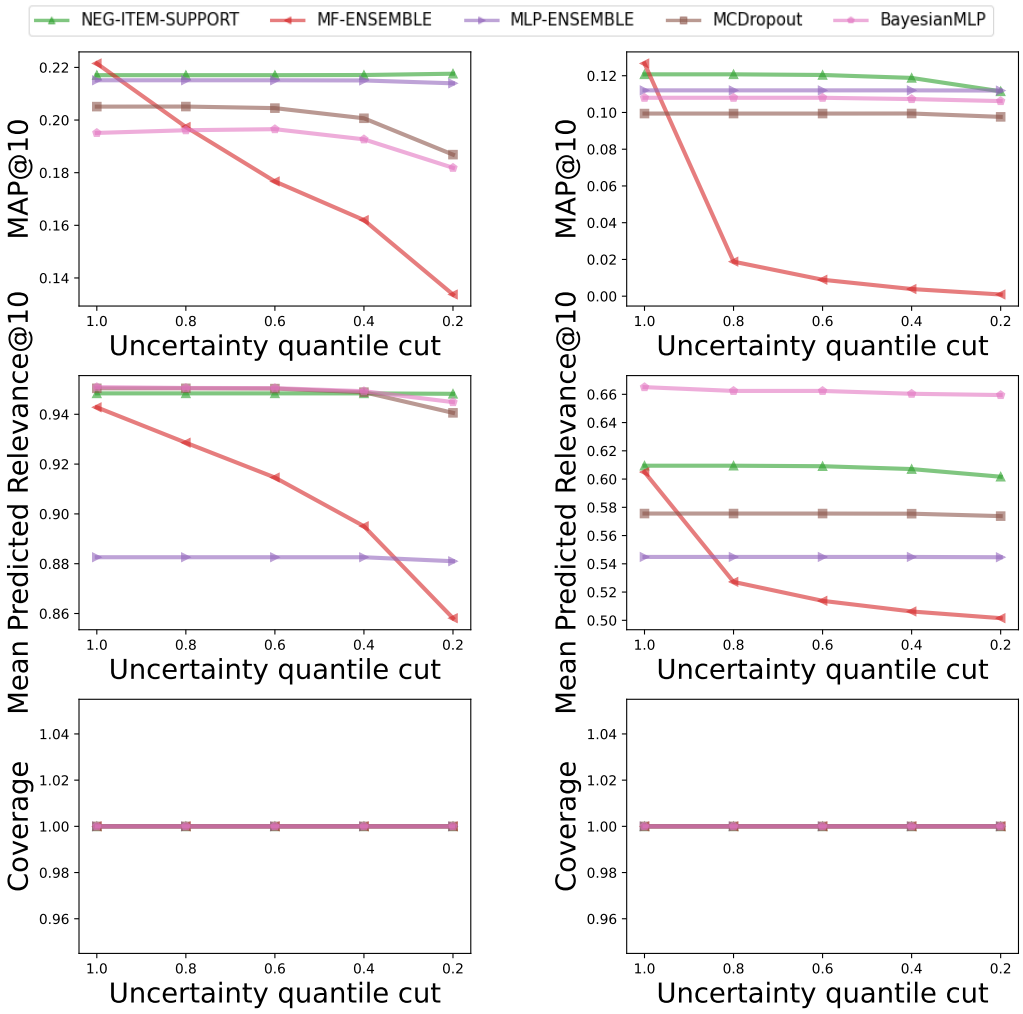
The results are shown in Figure 4.4. For every estimator, and every filtering threshold, we present the MAP@10 (Eq. 2.20), the Mean Predicted Relevance@10 (Eq. 4.10) and the Average Coverage (Eq. 2.27).

First, we observe that in no case does MAP increase with the use of UBF. In addition, in both datasets, we observe that the MAP for some models progressively decreases as τ grows: among those, MF-ENSEMBLE is the most noticeable. Interestingly, MF-ENSEMBLE is the only method whose Mean Predicted Relevance also decreases considerably when the cutting threshold is increased. This steep decrease in relevance is a possible explanation of why its MAP decreases so drastically. We do not observe any loss of coverage in any case.

With the exception of MF-ENSEMBLE, we note that in every other case, and in both datasets, the MAP and Mean Predicted Relevance varies very little, even when 80% of the candidate interactions are removed. From this behaviour, we postulate that the recommendations are not changing much after filtering. That is, recommendations given using UBF are similar to the traditional ReBR recommendations. This would happen only if the majority of the items recommended with ReBR have low uncertainty.

To verify this, we observe the relationship between relevance and uncertainty. In Figure 4.5, we illustrate the distribution of relevance and uncertainty for a randomly sampled set of interactions, for each uncertainty estimator, in the MovieLens dataset[9]. Notice, in every case except MF-ENSEMBLE, the negative correlation between relevance and uncertainty. This explains why UBF does not impact recommendations very strongly: uncertain interactions are not being recommended with ReBR in the first place. Again, MF-ENSEMBLE is an ex-

[9]We omit the corresponding figures for Pinterest, where we observed the same behaviour.



(a) MovieLens dataset.

(b) Pinterest dataset.

Figure 4.4: UBF: MAP, Mean Predicted Relevance and Coverage. This is for top-10 recommendations with increasing τ to filter different quantiles.

ception, but unfortunately, it also appears unsuccessful due to low URI and the loss of MAP on URI.

Therefore, we conclude that none of the methods explored in this chapter are particularly successful at uncertainty estimation in IRSs although we note the accuracy uplift observed with some of them.

4.6 Conclusions and Limitations

In this chapter, we explored methods for uncertainty estimation for implicit feedback-based recommender systems. In comparison to the explicit feedback case (see Chapter 3), uncertainty estimation for IRSs was even less explored

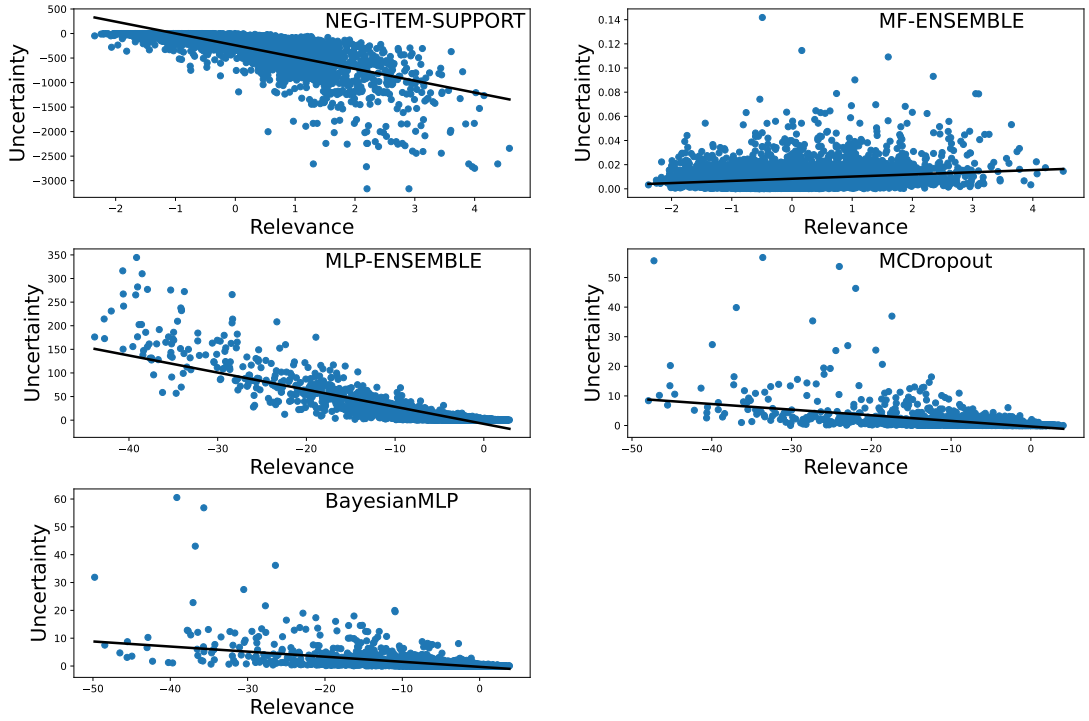


Figure 4.5: The distribution of relevance and uncertainty estimates on the MovieLens dataset. Each point represents a randomly sampled interaction. The black lines are least squares tendency lines.

prior to our work. We decided to emphasize neural network-based RSs because of their prominence in the RSs field.

The main takeaways from this chapter are:

- We confirm the superiority of MF over MLP — at least for these datasets. Even though this has already been previously shown by Rendle et al. [RKZA20], our methods (e.g. our data-splitting) differs from theirs. Therefore, our results give stronger support to this finding.
- We found that, in some cases, our systems that include uncertainty estimators have a higher recommendation accuracy then their respective MF or MLP baselines. The improvement is especially remarkable with the ensemble-based methods, but is also observed with MCDropout and BayesianMLP.
- In addition, MF-ENSEMBLE was also one of the top contenders when it came to correlation between accuracy and uncertainty, suggesting that this method can help to identify which users are prone to receive the most or least accurate recommendations.

- All the previous remarks highlight positives, but in all fairness all of our methods fail at their foremost goal of uncertainty estimation according to our metrics. We did not find evidence to support the usefulness of the uncertainty estimators in the chapter, especially in the top-n recommendation task. With that said, it is true that the failure of the estimators may be partially due to the metrics' issues, as we pointed in the case of URI.

In Chapter 6, we will discuss possible future work on the estimation of prediction uncertainty in IRSs. Our discussion there will partially build upon the following limitations that we identify in the work of this chapter.

- We limited this chapter to MF and MLP models. Although we expect our findings regarding the estimation of prediction uncertainty in IRSs to extend to other models, we can not guarantee they generalize to them all. In particular, there is a plethora of neural recommenders in the latest literature. We cannot be sure that our findings generalize to these various neural architectures.
- As in Chapter 3, we once more used a single train/test split, for reasons we already explained, and restricted our experiments to only two datasets. Therefore, although our results appear quite conclusive, it could be beneficial to extend our experiments to more datasets, particularly in different recommendation domains.
- We also acknowledge the limitation of our prediction uncertainty evaluation. In particular, the URI metric that we utilize is based on the relation between relevance and uncertainty. But again, we point to the disconnect between relevance and observation: not all unobserved interactions are irrelevant. Therefore, we believe the literature could strongly benefit from novel evaluation metrics, or at least a meticulous investigation of the existing metrics.

Chapter 5

Label Uncertainty in Implicit Feedback-Based Recommender Systems

Learning a model for a Implicit feedback-based Recommender System generally involves assuming that unobserved interactions are equally relevant to each other and less relevant than observed interactions (see Section 2.3.3). On the one hand, this general assumption is necessary to furnish the training dataset with negative samples. On the other hand, it is well known that observations are not an oracle for user preferences. For instance, there may be items that are relevant to a user but which the user has not consumed simply because the user is not aware of them, or due to other barriers, such as high price or limited availability. Users may also actually dislike some of the items they interacted with. This uncertainty regarding the true relevance of interactions is what we refer as *label uncertainty*.[1]

Noting this disconnect between user preferences and observed interactions, Hu et al. use additional information about the observed interactions to signify levels of confidence [HKV08]. For example, in a song recommender, the confidence may derive from the number of times each song has been played by the user.

While Hu et al. allow observed interactions to have varying levels of confidence, they still treat all the unobserved interactions as equally irrelevant – in the example above, all unobserved interactions refer to songs that the user has played

[1]This focuses on label uncertainty, and not prediction uncertainty. For this reason, in this chapter, where we use the word 'uncertainty' on its own, it is label uncertainty that we are referring to.

zero times. However, as explained above, there are several reasons why a user might not interact with a potentially relevant item. Therefore, the assumption of homogeneous irrelevance is not compelling. In practice, there should also be varying levels of confidence associated with each unobserved interaction, including those used as negative samples – that is, how plausible it is that the negative sample is indeed correctly labeled as negative. Mislabeled instances may impact recommendations drastically, especially for items that are not popular: these are almost always labeled as negatives [SYN+20].

Label uncertainty is very different from prediction uncertainty, which we have explored in Chapters 3 and 4. We defined prediction uncertainty as the expected imprecision of a prediction – in other words, the stochasticity of the model's *output*. Now, label uncertainty is the imprecision of the interaction labeling mechanism, which serves as *input* for the learning of recommendation models.

Note that even though prediction uncertainty and label uncertainty are defined differently, there is a connection between them. It is likely that label uncertainty to which a model is subject will manifest as prediction uncertainty in the model's output.

In Chapter 4, we directly tacked the problem of prediction uncertainty estimation in IRSs. Note that, in that chapter, the uncertainty estimators are based upon the output of the relevance model. For example, MF-ENSEMBLE's uncertainty follows from averaging several versions from the same uncertainty estimator. None of the methods in that chapter involve the direct *learning* of an uncertainty estimator[2]. This is for a good reason: in IRSs, unobserved interactions – that is, those whose uncertainty we want to predict – are part of the training set due to the use negative samples. This is paradoxical from the point of view of predicting uncertainty. There is no sense in learning to predict the uncertainty of an interaction that is part of the training set.

So, while we observe that the use of negative samples in IRSs limits our ability to estimate prediction uncertainty, we find that there is another form of uncertainty in IRSs that can be measured and exploited: label uncertainty. In this chapter, we discuss this form of uncertainty in depth.

We note in advance that this labeling problem is of especial concern for items in the long tail – that is, the items least interacted with. Because these items

[2]To clarify, although the methods in Chapter 4 are all doing learning, they are learning F (the relevance model), they are not learning G (the uncertainty model): Eqs. 4.4 and 4.7 show that G is computed from F.

have few observed interactions, they are assumed irrelevant to almost every user. Naturally, some or many of these assumed irrelevant interactions will be mislabeled. Therefore, estimating label uncertainty may be particularly useful for these long-tail items.

The only piece of work that we are aware of that models label uncertainty in IRS is by Wang et al. [WFZ+22]. Their approach, which they call Aleatoric Uncertainty-aware Recommendation (AUR), estimates the label uncertainty associated with each given relevance prediction. They show that, by estimating label uncertainty, it is possible to design an IRSs that is (i) more accurate and (ii) less prone to popularity bias.

Building on Wang et al.'s work, in this chapter we propose new methods for *learning* label uncertainty estimators in Implicit feedback-based Recommender Systems. We begin with an extensive review of Wang et al.'s AUR (Section 5.1). Then, we present alternatives to AUR (see Sections 5.2.1, 5.2.2, 5.3.2 and 5.3.3). These alternatives aim to fix some shortcomings that we observe in AUR. Finally, Section 5.5 presents the results of an empirical study, comparing AUR against the novel methods that we propose.

5.1 Aleatoric Uncertainty-aware Recommendation

The method proposed in [WFZ+22] is the first to tackle the labeling issue in IRS from a predictive perspective. Their proposal, named Aleatoric Uncertainty-aware Recommendation (AUR), is a model-agnostic learning procedure that allows for a vast class of Collaborative Filtering models to be employed for estimation of label uncertainty in IRS.

5.1.1 The AUR model

The AUR method is a point-wise learning procedure (see Section 2.3.3.1) that assign a numeric value to the implicit feedback data. Identically to Eq. 2.8, AUR sets $Y_{ui} = 1$ if user u has interacted with item i and $Y_{ui} = 0$ otherwise. In other words, the observed relevance of the interaction is set to the numeric value of 1 (0) if the interaction is observed (not observed). Then, by assuming $Y_{ui} \sim N(\mu_{ui}, \sigma^2_{ui})$, the likelihood of a given interaction Y_{ui} is given by,

$$\mathbb{P}(Y_{ui}|\mu_{ui},\sigma^2_{ui}) = N(Y_{ui}|\mu_{ui},\sigma^2_{ui}) = \frac{1}{\sigma_{ui}\sqrt{2\pi}} \exp\left[-\frac{(\mu_{ui}-Y_{ui})^2}{2\sigma^2_{ui}}\right] \tag{5.1}$$

Interestingly, AUR closely resembles the formulation of CPMF (Eq. 3.11).

The choice of the Gaussian distribution is compelling, due to this distribution being parameterized by a tendency and a dispersion measure, which can be understood as the average relevance of the interaction and its uncertainty. On the other hand, this model disrespects the binary nature of implicit feedback by treating it as a continuous random variable[3]. Later in this chapter, we will present alternative approaches that respect the binary nature of implicit feedback.

The negative log-likelihood, used to learn AUR, is,

$$-\log P(Y_{ui}|\mu_{ui},\sigma^2_{ui}) \propto \frac{(\mu_{ui}-Y_{ui})^2}{2\sigma^2_{ui}} + \log(\sigma_{ui}) \tag{5.2}$$

From Eq. 5.2, we note that AUR unravels the interaction relevance estimation problem into two: estimating the distribution's mean and variance, μ and σ^2, respectively. A mean prediction model $F_\theta : U \times I \to \mathcal{R}$ is defined, such that $\mu_{ui} = F_\theta(u,i)$, where θ is the model's parameter set. Similarly, a variance prediction model $G_\phi : U \times I \to \mathcal{R}$ is defined, parameterized by ϕ. From here onwards, for ease of notation, we omit the subscripts θ and ϕ. Also, following from Table 1.4, we write $F_{ui} = F(u,i)$ and $G_{ui} = G(u,i)$.

Now, the variance parameter σ^2_{ui} must be positive. To ensure this property, Wang et al. [WFZ+22] use an exponential form $\sigma^2_{ui} = e^{(G_{ui})}/K$, where K is a scale hyperparameter. In addition, they propose to use a regularizing prior for the variance $G_{ui} \sim N(0,\tau^2)$. Therefore, the posterior negative log-likelihood of the interaction variance is,

$$-\log \mathbb{P}(G_{ui}|Y_{ui},F_{ui}) \propto \frac{(F_{ui}-Y_{ui})^2}{\frac{2}{K}e^{G_{ui}}} + \frac{1}{2}G_{ui} + \frac{G^2_{ui}}{\tau^2} \tag{5.3}$$

Following from Eq. 5.1, and combined with the prior variances, after some simplification, the loss function for a batch of training instances $\mathcal{B}$ is given by,

[3]These random variables have two possible values according the training set (0 or 1) – and therefore should be discrete. But, AUR treats them as continuous: at prediction time, the values are not restricted to $\{0,1\}$.

$$\mathcal{L}(\mathcal{B}|\Theta) = \frac{1}{\#\mathcal{B}} \sum_{(u,i)\in\mathcal{B}} \frac{(F_{ui} - Y_{ui})^2}{e^{G_{u,i}}} + \beta G_{u,i} + \gamma G_{u,i}^2 + \lambda_\theta ||\theta|| \tag{5.4}$$

where $\Theta = \theta \cup \phi$ is the set containing all the parameters that compose $\mu_{ui}, \sigma_{ui} \forall\, u \in U, i \in I$, and β, γ are hyper-parameters. The additional term $||\theta||$ regularizes the relevance estimator, with regularization strength controlled by λ_θ.

5.1.2 Learning AUR

Eq. 5.4 offers a general loss function that allow models F and G to be learned. That is, given two parametric collaborative filtering models F_θ and G_ϕ, their parameters θ and ϕ can be learned simultaneously by stochastic gradient descent on the negative log-likelihood. Nevertheless, Wang et al. highlight that F and G tend to be strongly correlated: the higher the training error on learning an interaction relevance, the larger the uncertainty will be. In other words, the learner can select for higher uncertainty as a 'shortcut' to account for a poor relevance fit. For this reason, learning F and G simultaneously may lead to an inability to learn the best relevance estimator.

Hence, [WFZ+22] propose to learn AUR's F and G sequentially. Their proposal consists of two steps: first, to learn F_θ, the variance of every instance is fixed $\sigma_{ui}^2 = 1$; then, to learn the uncertainty estimator G_γ, with the parameters θ fixed to the values learned in the first step.

Wang et al. compare their sequential learning method to the simultaneous learning approach on two datasets, measuring Recall@50. Interestingly, in many cases, confounding expectations, the simultaneous learning method outperformed the sequential method. Moreover, they found that the optimal learning method depended on the underlying CF model. In particular, sequential learning was slightly outperformed by simultaneous learning when using a matrix factorization model (Eq. 2.1).

Overall, the differences in performance observed between sequential and simultaneous learning in [WFZ+22] were small and not decisive. That is, there is no empirical evidence to support the claim that the success (or otherwise) of AUR is related to the sequential learning algorithm introduced by Wang et al. For this reason, in our experiments, we will stick to simultaneous learning in all cases.

5.1.3 Top-n recommendations with AUR

AUR allows us to score candidate items. There is a novel form of ranking that may be used with AUR, referred to as Mislabeling-Aware Ranking (MAR), described below. It includes Relevance-Based Ranking (ReBR) (as in Chapter 4) as a special case. We note that other forms of ranking such as Uncertainty-Based Filtering (UBF) and Probability-of-Relevance Ranking (PRR) do not make sense with AUR. UBF discards the most uncertain candidates. In the case of AUR, this would result in us discarding candidates (items that are recorded as being unobserved) that are most likely to have been mislabeled, and therefore potentially relevant. As for PRR, it favours candidates with high probability of being relevant. As in Chapter 4, it is not applicable to AUR because AUR's scoring method is relevance-based by default.

Let us turn now to the AUR scoring function. Given that AUR estimates label uncertainty in addition to relevance, it is possible to combine these to give mislabeling-aware recommendations. The form of ranking that we denote Mislabeling-Aware Ranking (MAR), introduced by Wang et al., takes the uncertainty into account when ranking the candidate items.

Given a weighting parameter $\lambda \in [0, 1]$ (a hyper-parameter), the method scores user-item interactions according to the following rule,

$$\lambda F_{ui} + (1 - \lambda)\sqrt{G_{ui}} \tag{5.5}$$

The lower the value of λ, the higher the importance given to the label uncertainty. Note that if $\lambda = 1$, we obtain ReBR.

Using lower values of λ to favour uncertain interactions make sense for this form of uncertainty. The set of recommendation candidates for a user is typically drawn from the unobserved interactions for that user. Therefore, the higher their uncertainties, the more plausible it is that the interactions should not be recorded as being unobserved, and therefore are relevant.

In addition, in our opening remarks to this chapter, we noted the importance of label uncertainty estimation for long-tail items. According to Wang et al. recommending through MAR should increase recommendation accuracy on long-tail items. Because these items are infrequently interacted with, they tend to have higher label uncertainty. Therefore, by decreasing λ, we expect to favour

the recommendation of long-tail items. If so, MAR may also assist to reduce the popularity bias, which is a well-known issue in many recommendation algorithms [SYN+20].

5.2 The choice of F and G in AUR

As previously stated, AUR offers a general loss function (see Eq. 5.4) that allows for a vast family of choices for the relevance and uncertainty estimators, F and G respectively. In principle, both can be any parameterized CF model.

In the original paper [WFZ+22], Wang et al. employ Matrix Factorization (MF) (Eq. 2.1), LightGCN [HDW+20] and VAE [LKHJ18], motivated, it seems, only by the popularity of these models. As we already mentioned, their experiments included variants in which F and G were learned either sequentially or simultaneously. Surprisingly, it was the simplest model, designated MF-AUR-J, that achieved the highest overall recall in both datasets that they analyzed. MF-AUR-J uses MF and learns F and G simultaneously (or jointly, J).

In our empirical study (Section 5.4), we include MF-AUR-J – simply referred to as AUR there. Here, we propose two new alternatives: Implicit CPMF and Gaussian-MLP.

5.2.1 Implicit CPMF

The fact that Wang et al.'s simplest model achieved their best results, led us to question how complex G must be in order to accurately estimate label uncertainty. Moreover, we saw a similar picture in Chapter 3, where CPMF, an architecture that models each user and item uncertainty with a single parameter, proved better than models with many more parameters.

As one much simpler alternative to AUR, and following the same principles, we introduce a new model that we name ICPMF (Implicit CPMF). For this model, we define G as a simple product between a user variance parameter and an item variance parameter. F on the other hand is a traditional MF model. That is,

$$F_\theta(u, i) = p_u^T q_i \tag{5.6}$$

$$G_\phi(u, i) = \sigma_u \sigma_i \tag{5.7}$$

where σ_u and σ_i are learnable scalar parameters for each user and item, respectively, and p_u, q_i are user and item latent factors as in Eq. 2.1. To ensure positivity of the Gaussian variance, we use the same transformation as AUR, $\sigma^2_{ui} = e^{\sigma_u \sigma_i}/K$. The model parameters – that is, all p_u, σ_u for $u \in U$ and q_i, σ_i for $i \in I$ – are learned simultaneously through stochastic gradient descent on the negative log-likelihood (Eq. 5.4), exactly as in AUR.

5.2.2 Gaussian-MLP

The two methods introduced so far, MF-AUR-J and ICPMF, tackle relevance and label uncertainty estimation as a two model problem: F and G are two separate models, learned simultaneously. But in this section, we propose a novel approach that unifies F and G using a two-output neural network. This method, that we denote Gaussian-MLP, serves the same purpose as MF-AUR-J and ICPMF, but with the particularity of defining a single neural network-based model that replaces F and G.

We define a two-output neural network, as follows. We start with a latent factor model, where users and items are model by d-sized embeddings p_u and q_i, respectively. Then, similarly to the MLP (Eq. 2.3), we define a feed-forward neural network that takes the interaction's concatenated embeddings. The final layer of this feed-forward neural network will output a pair $\langle \mu_{ui}, \sigma^2_{ui} \rangle$. More precisely, our proposed network comprises a set of feed-forward layers $f_1, \ldots, f_L$ defined as,

$$f_0 = p_u || q_i \tag{5.8}$$

$$f_l = \text{ReLU}(W_l f_{l-1}), \quad \text{for } l \in 1, \ldots, L-1; \tag{5.9}$$

$$f_L = \langle w^t_{L\mu} f_{L-1}, g(w^t_{L\sigma} f_{L-1}) \rangle \tag{5.10}$$

The network parameters are: W_l, the weight matrix for hidden layer l, for $l = 1, \ldots, L-1$; $w_{L\mu}$, the output layer's weight vector for the relevance output; and $w_{L\sigma}$, the output layer's weight vector for the uncertainty output. For the uncertainty output, activation function g is used to ensure positivity. In this work, we employ a scaled exponential form $g(x) = e^x/K$, where K is a scaling hyper-parameter.

Thus, the full set of parameters for the model is $\Theta = \{\{p_u\}_{u \in U}, \{q_i\}_{i \in I}$ $\{W_l\}_{l \in 1,\ldots,L-1}, w_{L\mu}, w_{L\sigma}\}$. It is straightforward to learn these parameters using stochastic gradient descent and same loss function that AUR uses (Eq. 5.4).

Note that, by using AUR's loss function, we assume that μ_{ui} and σ^2_{ui} are the parameters of a Gaussian distribution, as AUR does. That is, $Y_{ui} \sim N(\mu_{ui}, \sigma^2_{ui})$.

5.3 Binary Mislabeling-aware Recommenders

The models described in the previous two sections (AUR, Implicit CPMF and Gaussian-MLP) all share the same assumption of normality $Y_{ui} \sim N(\mu_{ui}, \sigma^2_{ui})$. However, by making this assumption, these models treat implicit feedback data as continuous – that is, they are assumed to be numerical values assuming 0 (non-observed interaction) or 1 (observed interaction). In Section 2.3.3.1, we argued that a more sound approach is to predict each interaction's probability of observation $P(Y_{ui} = 1)$. We can say that the formulation used by these three models use a continuous variable to approximate the (true) binary implicit feedback variable Y_{ui}.

In this section, we develop two novel methods for learning mislabeling-aware systems. They are shaped by the same motivation behind AUR. But both methods respect the binary nature of implicit feedback: they tackle the problem of directly estimating the interaction probabilities $P(Y_{ui} = 1)$. Before presenting these two new methods, we present a discussion of the use of implicit feedback that serves several purposes: it deepens our reasoning about mislabeling uncertainty; it critiques AUR; and it motivates the two new methods that follow.

5.3.1 Rethinking implicit feedback data

Implicit feedback data consists of a set of observed interactions R between a set of users U and a set of items I. The goal of an IRS is to utilize the knowledge in R to infer the relevance of non-observed interactions $\{u, i\} \notin R$. Formally, let $Y_{ui} = 1$ denote the event that user u has interacted with item i. Then, traditional IRSs (see Section 2.3.3) reduce the problem of relevance inference to estimating the probability of future interactions $P(Y_{ui} = 1 | u, i \in R^-)$.

Now, as previously noted, there is a disconnect between relevance and observations. First of all, it is likely that every user has disliked some items they interacted with. That is, if we let $\zeta_{ui} = 1$ denote the event that user u likes item i – or, synonymously, item i is relevant to user u —, then, our reasoning implies that $\mathbb{P}(\zeta_{ui} = 1 | Y_{ui} = 1) \neq 1$.

By the same reasoning, and of much greater importance, is that $\mathbb{P}(\zeta_{ui} = 0 | Y_{ui} =$

$0) \neq 1$. In fact, this is what motivates IRSs: discovering those interactions that are labeled as 0 at training time that are the most likely to be relevant.

It is sensible to assume that interactions have a certain probability of being observed, and that this probability is guided by their relevance $\mathbb{P}(Y_{ui} = 1|\zeta_{ui})$. In addition, the interaction probability also depends on exposure. Let O_{ui} be an exposure indicator, that is $O_{ui} = 1$ if user u has been exposed to item i, and 0 otherwise. One heuristic, which can be found, for example, in Liang et al.'s ExpoMF model [LCMB16] is $O_{ui} = 0 \implies Y_{ui} = 0$ – in other words, users must be aware of an item, prior to consuming it.

Saito et al. couple relevance and exposure [SYN+20]. They propose to model the observations as,

$$\mathbb{P}(Y_{ui} = 1) = \mathbb{P}(\zeta_{ui} = 1)\,\mathbb{P}(O_{ui} = 1) \tag{5.11}$$

According to Eq. 5.11, interactions have a certain probability of being observed, and that probability is proportional to both the probability of relevance and the probability of exposure. Turning our attention back to AUR, we observe that,

$$F_{ui} \propto \mathbb{P}(Y_{ui} = 1) \tag{5.12}$$

$$G_{ui} \propto \mathbb{P}(Y_{ui}^{actual} \neq Y_{ui}) \tag{5.13}$$

where Y^{actual} denotes the actual relevance indicator, that is not subject to mislabeling, and the symbol $\propto$ denotes direct proportionality.

We note that, if $(u, i) \in R^-$, then $\mathbb{P}(Y_{ui}^{actual} \neq Y_{ui})$ indicates how likely it is that the interaction is mislabeled as negative. If this probability is high, then it is plausible that the user is unaware of the item – that is, $O_{ui} = 0$ –, because if the user were aware of an item that was relevant to them, they would very likely interact with it.[4] Hence, for $(u, i) \in R^-$, we suppose that $G_{ui} \propto \mathbb{P}(O_{ui} = 0)$. Reordering Eq. 5.11, we have,

[4]We say plausible because mislabeling may have causes other than lack of awareness.

$$\mathbb{P}(\zeta_{ui} = 1) = \frac{\mathbb{P}(Y_{ui} = 1)}{\mathbb{P}(O_{ui} = 1)} \tag{5.14}$$

Now, we can compare Eq. 5.14 to Eq. 5.5 – $\lambda F_{ui} + (1 - \lambda)\sqrt{G_{ui}}$ –, which is the scoring function that AUR uses for Mislabeling-Aware Ranking (MAR). Because $F_{ui} \propto \mathbb{P}(Y_{ui} = 1)$ and following our assumption that $G_{ui} \propto \mathbb{P}(O_{ui} = 0)$, we observe that there is a strong connection between MAR and the modeling principle proposed by Saito et al. (Eq. 5.11). The relevance probability $\mathbb{P}(Y_{ui} = 1)$ grows with the probability that the item is liked, $P(\zeta_{ui} = 1)$. Hence, so does MAR's score, because F_{ui} is directly proportional to $\mathbb{P}(Y_{ui} = 1)$. Similarly, the relevance probability $\mathbb{P}(Y_{ui} = 1)$ also grows with the observation probability $\mathbb{P}(O_{ui} = 1)$. Therefore, so does MAR's score, because G_{ui} is *inversely* proportional to $\mathbb{P}(O_{ui} = 1)$.

We conclude that MAR is sound, in particular due to its similarity to the observation model from Eq. 5.11. However, AUR (the method that motivates MAR) has the flaw of treating implicit feedback as numerical data in $\{0, 1\}$. For this reason, F_{ui} is not an estimate for $P(Y_{ui} = 1)$, but instead a correlated quantity. It is not clear whether this flaw compromises recommendation accuracy.

In the next two sections, we propose two novel models that respect the binary nature of the implicit feedback and explicitly estimate $P(Y_{ui} = 1)$. Our goal is to asses whether these alternative methods, which do not share the aforementioned flaw with AUR, will achieve higher recommendation accuracy.

5.3.2 Gaussian binary ranking

A standard principle for learning an RS model for implicit feedback data is to treat the problem as one that involves binary classification and to predict, for each interaction, the probability that it is relevant; we see this, for example, in approaches that use Binary Cross-Entropy (Eq. 2.10). Inspired by this, we introduce Gaussian Binary Ranking (GBR): a classification oriented method to estimate probabilities of relevance and the uncertainties associated with them.

Let $r_{ui} \sim \mathcal{N}(\mu_{ui}, \sigma^2_{ui})$ be a Gaussian variable that models the relevance of item $i \in I$ to user $u \in U$. The probability that the user u interacts with item i is modeled as,

$$\mathbb{P}(Y_{ui} = 1) = \mathbb{P}(r_{ui} > 0) = \mathbb{P}(\mathcal{N}(\mu_{ui}, \sigma^2_{ui}) > 0) = 1 - \Phi\left(\frac{-\mu_{ui}}{\sqrt{\sigma^2_{ui}}}\right) \tag{5.15}$$

where Φ is the standard Gaussian cumulative distribution function. Naturally, the probability for a pair to be unobserved is,

$$\mathbb{P}(Y_{ui} = 0) = \mathbb{P}(r_{ui} \leq 0) = \Phi\left(\frac{-\mu_{ui}}{\sqrt{\sigma^2_{ui}}}\right) \tag{5.16}$$

Therefore, the likelihood of a given interaction is given by,

$$\mathbb{P}(Y_{ui}|\mu_{ui}, \sigma^2_{ui}) = \mathbb{P}(Y_{ui} = 1)^{\delta_{ui}} \mathbb{P}(Y_{ui} = 0)^{1-\delta_{ui}} \tag{5.17}$$

where $\delta_{ui} = 1$ if u has interacted with i, and 0 otherwise. Notice that, although we employ the notation $Y_{ui} = 1$ and $Y_{ui} = 0$, these values are only labels, indicating whether or not the user-item interaction was observed, and not numeric values, as they were in the case of AUR.

The negative log-likelihood for one interaction is,

$$-\log \mathbb{P}(Y_{ui}|\mu_{ui}, \sigma^2_{ui}) = -\delta_{ui} \log\left(1 - \Phi\left(\frac{-\mu_{ui}}{\sqrt{\sigma^2_{ui}}}\right)\right) - (1 - \delta_{ui}) \log \Phi\left(\frac{-\mu_{ui}}{\sqrt{\sigma^2_{ui}}}\right) \tag{5.18}$$

Now, we borrow from AUR. We define two models $F_\theta(u, i)$ and $G_\phi(u, i)$ such that $\mu_{ui} = F_{ui}$ and $\sigma^2_{ui} = e^{G_{ui}}/K$. Additionally, we employ the same regularizing prior $G_{ui} \sim N(0, \tau^2)$. It follows that the posterior negative log-likelihood of the variance is,

$$\begin{aligned} -\log \mathbb{P}(G_{ui}|Y_{ui}, F_{ui}) \propto & -\delta_{ui} \log\left(1 - \Phi\left(\frac{-\mu_{ui}}{\sqrt{e^{G_{ui}}/K}}\right)\right) - \\ & (1 - \delta_{ui}) \log \Phi\left(\frac{-\mu_{ui}}{\sqrt{e^{G_{ui}}/K}}\right) + \frac{G_{ui}}{\tau^2} \end{aligned} \tag{5.19}$$

GBR is, arguably, more sound than AUR since it handles implicit feedback more

authentically. But, as soon as we began our experimentation with it, we noticed an identifiability problem. Note that, by construction, we have $\mathbb{P}(Y_{ui} = 0) = \mathbb{P}(r_{ui} \leq 0)$. Note also that the probability density function of r_{ui} is symmetrical around its average μ_{ui}.[5] As a consequence, if $\mu_{ui} = 0$ we have,

$$\mathbb{P}(Y_{ui} = 0) = \mathbb{P}(r_{ui} \leq 0) = \mathbb{P}(r_{ui} > 0) = \mathbb{P}(Y_{ui} = 1) \tag{5.20}$$

where the second equality follows from the symmetry around 0 (the assumed value of μ_{ui}). Note that, the equation above holds regardless of the σ^2_{ui} value. This causes σ^2_{ui} to be unidentifiable in the case of $\mu_{ui} = 0$. In practice, during learning, this causes the uncertainty of interactions with average relevance in the vicinity of 0 to not converge.

For this reason GBR is not able to provide proper label uncertainty estimates for every instance. We chose to leave the theoretical formulation of the method here, since it was an important step in the development of our next model (Section 5.3.3). But we did not include it in our experiments.

5.3.3 Hierarchical binary ranking

In this section, we propose Hierarchical Binary Ranking (HBR), a hierarchical Bayesian model as an alternative to GBR. Like GBR, it defines Gaussian latent relevance variables r_{ui}. The choice of the Gaussian distribution seems appropriate, because it is parameterized by the two quantities of interest to us – the average relevance and the uncertainty. Where HBR differs from GBR is in the way it connects the latent continuous relevances to the implicit feedback. GBR uses a straightforward connection (Eq. 5.15), but this is also responsible for GBR's identifiability issue. HBR employs a more sophisticated form of connection (Eq. 5.21) that turns out to not suffer from the identifiability problem.

5.3.3.1 The HBR model

Let $r_{ui} \sim \mathcal{N}(\mu_{ui}, \sigma^2_{ui})$ be a latent relevance score for user u's interaction with item i, and let $Y_{ui} = 1$ denote the event that user u has interacted with item i. We propose a novel method, named Hierarchical Binary Ranking (HBR), that models such probabilities using the standard logistic function,

[5] This is a property of the Gaussian distribution.

$$\mathbb{P}_{ui} = \mathbb{P}(Y_{ui} = 1 | r_{ui}) = \frac{1}{1 + e^{r_{ui}}} \tag{5.21}$$

In other words, we say,

$$Y_{ui} | r_{ui} \sim \text{Bernoulli}\left(\frac{1}{1 + e^{r_{ui}}}\right)$$

Our proposal casts the scoring problem into the task of estimating the parameters of each interaction's latent relevance distribution, similarly to AUR. The difference with AUR, as previously stated, is that we now offer an explicit estimator for $P(Y_{ui} = 1)$, acknowledging the binary nature of the implicit feedback.

We note that r_{ui} is a random variable, and therefore Eq. 5.21 defines a hierarchical Bayesian model [GCSR95]. That is, there are two random variables (for each interaction) Y_{ui} and r_{ui} that are connected through a dependency structure. More precisely, their joint distribution follows from the Bayes theorem,

$$\mathbb{P}(Y_{ui} \cap r_{ui}) = \mathbb{P}(Y_{ui} | r_{ui}) \mathbb{P}(r_{ui}) \tag{5.22}$$

Now, as in all the previous methods in the chapter, our goal is to parameterize μ_{ui} and σ^2_{ui}, the parameters of the (Gaussian) latent relevances r_{ui}, and subsequently learn the parameters from the implicit feedback data. In particular, we can follow the same approach as before and define a mean prediction model F_θ and a variance prediction model G_ϕ. The parameters θ and ϕ must now be treated as Bayesian parameters – that is, they must follow some prior probabilistic distribution. This is because HBR relies on approximate Bayesian inference, for reasons that will become clear shortly.

Because we must define a prior distribution for the parameters, it is cumbersome to formalize HBR for general parameterizations of F and G. For this reason, we opt to show the formal development with one particular parameterization that closely resembles ICPMF. More precisely, we will follow the same parameterization for F and G as ICPMF (Eqs. 5.6 and 5.7). This choice has two motivations: first, the simplicity of ICPMF's uncertainty estimator translates

into easier prior definitions, and secondly, ICPMF has shown good performance in our experiments, as will be shown in Section 5.5. We emphasise, however, that this parameterization is only one of many possible choices.

We assume normality for all the parameters in the model. Then the hierarchical model is as follows,

$$\begin{aligned}
\text{Stage I (Observations):}\, & Y_{ui}|r_{ui} \sim \text{Bernoulli}\left(\frac{1}{1+e^{r_{ui}}}\right) \\
\text{Stage II (Latent relevances):}\, & r_{ui}|p_u, q_i, \sigma_u, \sigma_i \sim \mathcal{N}(p_u^T q_i, e^{\sigma_u \sigma_i}) \\
\text{Stage III (Priors):}\, & p_u \sim \mathcal{N}^d(0, \alpha_u I_{d\times d}); q_i \sim \mathcal{N}^d(0, \alpha_i I_{d\times d}); \\
& \sigma_u \sim \mathcal{N}(0, \alpha_{\sigma u}); \sigma_i \sim \mathcal{N}(0, \alpha_{\sigma i}))
\end{aligned}$$

where $\mathcal{N}^d$ denotes a d-dimensional multivariate Gaussian.[6]

We let $Y = [Y_{ui}]$ denote the implicit feedback matrix, $Z = [r_{ui}]$ denote the latent interaction relevances, $P^{\#U\times d}$, $Q^{\#I\times d}$ denote the user and item embeddings matrices, respectively, and $\sigma_U^{\#U\times 1}$, $\sigma_I^{\#I\times 1}$ denote the variance parameter vectors for users and items, respectively. For ease of notation, we write the latent parameter set $\Theta = \{P, Q, \sigma_U, \sigma_I\}$.

The joint probability distribution for the observed interactions is,

$$\begin{aligned}
\mathbb{P}(Y, Z, \Theta) &= \mathbb{P}(Y, Z|\Theta)\,\mathbb{P}(\Theta) \\
&= \mathbb{P}(Y|Z)\,\mathbb{P}(Z|\Theta)\,\mathbb{P}(\Theta) \\
&= \mathbb{P}(Y|Z)\,\mathbb{P}(Z|\Theta)\,\mathbb{P}(P)\,\mathbb{P}(Q)\,\mathbb{P}(\sigma_U)\,\mathbb{P}(\sigma_I) \qquad (5.23)
\end{aligned}$$

Eq. 5.23 makes evident the hierarchical structure of the model. The responses Y depend on the latent parameters in Θ only through the latent relevances Z. Figure 5.1 illustrates HBR as a probabilistic graphical model in the style of Chapter 8 of [Bis06].

[6] We abuse the notation by writing 0 as the multivariate distributions' average. In fact, the averages are d-dimensional zero vectors.

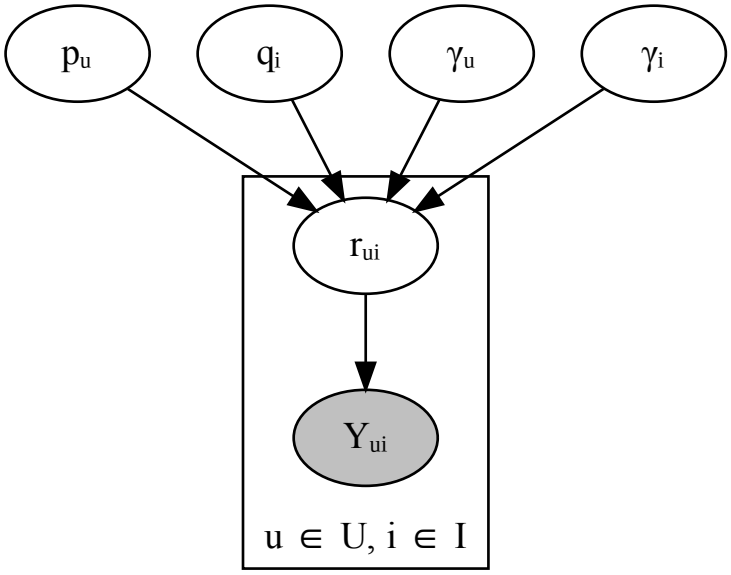


Figure 5.1: HBR's probabilistic graphical model. The white (non-filled) circles indicate non-observed (latent) random variables, while the grey (filled) circle indicates the observed variable – in this case, the binary implicit feedback. The arrows indicate probabilistic dependencies.

5.3.3.2 Learning HBR

The posterior distribution of the latent variables – that is, their distribution given the implicit feedback Y – can be expressed as,

$$\mathbb{P}(\Theta, Z|Y) = \frac{\mathbb{P}(Y, Z, \Theta)}{\int \mathbb{P}(Y, Z, \Theta) d\Theta dZ} \tag{5.24}$$

However, working with this posterior distribution is infeasible, because the denominator (i.e. the evidence) is intractable due to the multidimensional integral involved. For this reason, we must resort to using approximate inference.

The tools for approximate Bayesian inference problems such as this one revolve around Markov Chain Monte Carlo methods (MCMC) [Gey92] and Stochastic Variational Inference (SVI) [HBWP13]. In fact, MCMC has been previously explored for Explicit feedback-based Recommender Systems [SM08]. In particular,

CPMF has a Bayesian extension based on MCMC [WLW+18]. However, it is unclear how a similar MCMC algorithm could be employed in the case of IRSs due to the necessity of negative sampling.

For this reason, we use SVI. Unlike MCMC, which is a sampling method, SVI is an optimization method. For this reason, the learning can be performed in batches, which can include randomly-selected negative samples, as in Algorithm 1.

SVI consists in searching for a variational distribution $v(Z, \Theta|\eta)$ that serves as an approximation to the true posterior $\mathbb{P}(\Theta, Z|Y)$. In practice we pick a family of distributions parameterized by η, that are called variational parameters – that is, for each value of η, we recover a different distribution for the chosen family. Then, we choose η such that $v(Z, \Theta|\eta)$ is as close as possible to the true posterior $\mathbb{P}(\Theta, Z|Y)$. This distribution similarity is measured according to the Kullback-Leiber (KL) divergence. It can be shown that minimizing the KL divergence is equivalent to minimizing the Evidence Lower Bound (ELBO). In fact, the ELBO is equal to the negative KL divergence up to an additive constant, and is defined as,

$$\text{ELBO} := \mathbb{E}_{v(Z,\Theta|\eta)}[\log \mathbb{P}(\Theta, Z|Y) - \log v(Z, \Theta|\eta)] \tag{5.25}$$

Typically, the true posterior is very complex and is not contained in the variational family. Therefore, the quality of the approximated solution through SVI depends on the choice of the variation family over which we search for the optimal $v(Z, \Theta|\eta)$.

For HBR, we consider variational distributions that factorize similarly to the joint distribution Eq. 5.23. More precisely, we assume,

$$\begin{aligned} v(Z, \Theta|\eta) &= v_Z(Z|\eta) v_\Theta(\Theta|\eta) \\ &= v_Z(Z|\eta) v_P(P|\eta)\, v_Q(Q|\eta)\, v_{\sigma_U}(\sigma_U|\eta)\, v_{\sigma_I}(\sigma_I|\eta) \end{aligned} \tag{5.26}$$

where $v_\star$ denotes the variational distribution of the variable matrix $\star$. The choice for this family of factorizable distributions is in line with the construction of our priors, that lead to an also factorizable joint distribution (see Eq. 5.23).

At this point, it is useful to recap that, although we employ a full Bayesian treatment of our parameters – that is, Θ is a set of random variables rather than a set of deterministic parameters –, our goal is not to study the uncertainty around the parameters, but instead only the uncertainty of the relevances r_{ui} and probabilities of relevance Y_{ui}. For this reason, we can use point mass distributions (or delta distributions) for all $v_P, v_Q, v_{\sigma_I}, v_{\sigma_I}$. A point mass distribution is such that all the probability mass is concentrated around a single value, which is the distribution's parameter.

Notice that by choosing point mass distributions, the number of variational parameters is equal to the dimensions of the latent matrix. To give a concrete example, if P is a 10×10 matrix – that is, the dataset has 10 users and we employ latent embeddings of size 10 – then v_P is a delta distribution parameterized also by a 10×10 parameters matrix. In this example, the learned variational parameters of the delta distribution will be deterministic estimates for P. In the next section, we will see that the choice of point mass distributions also greatly reduces prediction time.

Learning the variational parameters η follows through stochastic gradient decent. On each step, the ELBO term (Eq. 5.25) is optimized for a data batch. We emphasise that each data batch contains positive and negative interactions, as per Algorithm 1.

Our implementation of HBR takes advantage of Automatic Differentiation Variational Inference (ADVI) [KTR+17], and is implemented in Pyro [BCJ+18].

5.3.3.3 Top-n recommendations with HBR

Our choice for point mass variational distributions leads to point estimates for the model parameters p_u, q_i, σ_u, σ_i. Therefore, even though HBR is based on variational inference, we do not need to do any posterior inference to make recommendations. Instead, we have a straightforward way to get point estimates for the average relevance $F_{ui} = \mu_{ui}$ and uncertainty $G_{ui} = \sigma^2_{ui}$. We simply follow Equations 5.6 and 5.7. Now, having μ_{ui} and σ^2_{ui}, we may follow, at least, the following two ranking criteria.

Ranking based on r_{ui}

Our first possible ranking criterion follows directly from MAR. Because HBR offers estimates for μ_{ui} and σ^2_{ui} (F_{ui} and G_{ui}), it can borrow MAR's ranking

criterion from Eq. 5.5. The reasoning is analogous to AUR: the value of λ controls (inversely) the importance given to label uncertainty – that is, the lower is λ, the higher is the importance given to G_{ui}, and therefore more emphasis is given to recommending potentially relevant items that are mislabeled during training. To facilitate referencing, we will use HBR-R to refer to the HBR model with MAR's ranking criterion[7]. Again, we note that ReBR is the special MAR case where $\lambda = 0$.

Ranking based on $\mathbb{P}_{ui}$

Above, we discussed ranking through the latent relevance (r_{ui}) parameters (μ_{ui}, σ^2_{ui}), intentionally leaving out the interaction probability $\mathbb{P}_{ui}$. Now, since we are aiming to treat implicit feedback as binary, let us examine the meaning of μ_{ui} and σ^2_{ui} in relation to $\mathbb{P}_{ui}$.

By construction, we have $r_{ui} \sim \mathcal{N}(\mu_{ui}, \sigma^2_{ui})$, and as a consequence, $\mathbb{P}_{ui} = \frac{1}{1+e^{r_{ui}}} \sim$ Logit-Normal$(\mu_{ui}, \sigma^2_{ui})$ [AS80]. Therefore, we can use some known results about the logit-normal distribution.

First, let s denote the standard logistic function. Then, $s(\mu_{ui})$ is the median of the interaction probability, $s(\mu_{ui}) = \text{Med}(\mathbb{P}_{ui})$ [HS22]. Therefore, μ_{ui} on its own is a robust point estimate for the interaction probability. Unfortunately, the logit-normal distribution has no analytical solution for any measure of dispersion, such as variance. But, as shown in [FL08], there is a positive correlation between the variance of $\mathbb{P}_{ui}$ and σ^2_{ui}.

Second, there are some known approximations for the mean and variance of $\mathbb{P}_{ui}$. Again, let s denote the standard logistic function. Then, following from [Dau17], one possible set of approximations is,

$$
\begin{aligned}
E[\mathbb{P}_{ui}] &= s\left(\frac{\mu_{ui}}{\sqrt{1+\frac{3}{\pi^2}\sigma^2_{ui}}}\right) \\
\text{Var}[\mathbb{P}_{ui}] &= s\left(\frac{\mu_{ui}}{\sqrt{1+\frac{3}{\pi^2}\sigma^2_{ui}}}\right)\left(1-s\left(\frac{\mu_{ui}}{\sqrt{1+\frac{3}{\pi^2}\sigma^2_{ui}}}\right)\right)s\left(1-\frac{1}{\sqrt{1+\frac{3}{\pi^2}\sigma^2_{ui}}}\right)
\end{aligned}
\tag{5.27}
$$

Based on these, we can design another ranking rule analogous to MAR. In this case, we rank items according to the following,

[7] We use the suffix R to reinforce that ranking is based on the latent relevances r_{ui}.

$$\lambda E[\mathbb{P}_{ui}] + (1 - \lambda)\text{Var}[\mathbb{P}_{ui}] \tag{5.28}$$

In our experiments, we will use HBR-Y to refer to a system that uses the HBR model but with the ranking rule from Equation 5.28[8].

Note again that, if $\lambda = 1$, then we obtain a ranking based only on $E[\mathbb{P}_{ui}]$ – a ranking that is parallel to ReBR. Nevertheless, we also note that $E[\mathbb{P}_{ui}]$ is a function of both μ_{ui} and σ_{ui}. Therefore, ranking based solely on $E[\mathbb{P}_{ui}]$ is already in a way accounting for the uncertainty in the latent relevances.

Before we finish, we make two final remarks. First, we emphasize that HBR-R and HBR-Y are the same model, the only difference is in the form of item ranking employed at the top-n recommendation stage. In the light of this, we will also abuse names a little: for simplicity in what follows, we say that HBR-Y, like HBR-R, uses MAR, although, in fact, only the latter uses MAR (Eq. 5.5); the former uses something that is similar to MAR (Eq. 5.28). Secondly, as noted by [Dau17], $\text{Var}[\mathbb{P}_{ui}]$ is a monotonically increasing function of σ_{ui}, and therefore HBR-Y should not differ too drastically from HBR-R.

5.4 Empirical Study

In this section, we compare the methods for mislabeling-aware ranking that were introduced in this chapter. In particular, our goal is to contrast the recommendation accuracy of the new models proposed in this chapter with AUR, the method taken from the literature.

As with previous chapters, we ensure reproducibility of the experiments by making all the code available in `github.com/vcoscrato/uncertain`.

5.4.1 Models

Our experiments comprise the following models: AUR, ICPMF, Gaussian-MLP and HBR – the last of these, in fact, being further subdivided into HBR-R and HBR-Y. With the exception of Gaussian-MLP, which is a unified model, all these models use estimators of relevance and of uncertainty, denoted F and G above.

[8] We choose the suffix Y because ranking in this case is based on the interaction probability $\mathbb{P}(Y_{ui} = 1|r_{ui})$.

In all cases, we use MF for F; and, in most cases, we use MF for G also. The exceptions to the latter are the uncertainty estimators G for ICPMF and for the HBR models, where, by design, G estimates uncertainty as the product of user and item latent variances (Eq. 5.7). Using the same underlying estimator for F and G wherever possible gives us a fair comparison.

Apart from seeing which of the models has the highest accuracy, the experiments should be revealing in two other ways. First, we may evaluate the necessity of the more complex (higher dimensional) uncertainty estimator employed by AUR against the much simpler one of ICPMF and HBR. In addition, these estimators of varying complexity are compared against Gaussian-MLP, a method that is rather different from the previous two because it unifies F and G.

Second, we can investigate the effect of giving a correct treatment to the implicit feedback, as discussed in Section 5.3. AUR, ICPMF and Gaussian-MLP approximate the implicit feedback with a numerical formulation whereas HBR (both HBR-R and HBR-Y) respect the binary nature of the implicit feedback, but learns an approximation for the true posterior through variational inference. In this respect, one particular comparison is of special interest: because we use the exact same relevance and uncertainty representation for both ICPMF and HBR[9], we are able to isolate this difference and can see its effect on the results.

We add to our results the best MF baselines for each dataset, according to Section 4.5.1. More precisely, what we refer to as MF (Baseline) in the following results is MF-BPR for the MovieLens dataset and MF-MSE for Pinterest.

5.4.2 Datasets

We use the same datasets as in Section 4.4.2 of Chapter 4. Moreover, we use the same data partitions as before.

5.4.3 Methods

5.4.3.1 Tuning

The models that we compare have a number of hyper-parameters that need to be tuned. To do so, we use a Bayesian parameter search, as per Section 4.4.3.2. With

[9]We remind the reader that we developed HBR to use the same relevance and uncertainty representation as ICPMF. But this was a choice. Variants of HBR, with other representations, can be derived.

each method, we learn 20 models with different values for the hyper-parameters, sampled as follows:

- For all models, the learning rate is sampled from [0.00001, 0.001]. The number of negative samples per positive interaction in each data batch is sampled from $\{1, 2, \ldots, 30\}$.
- For all models based on AUR's objective (Eq. 5.4) – that is, AUR, ICPMF and Gaussian-MLP – we sample β from $\{1e{-}3, 1\}$ and γ from $\{1e{-}4, 1e{-}1\}$.
- AUR and ICPMF: The L2 penalty factor applied to the user and item factors for the relevance estimator (λ_θ in Eq. 5.4) is taken from $[10^{-5}, 10^{-3}]$.
- Gaussian-MLP: Again, we use the same three-layer MLP as in [HLZ+17], with dropout rate during training taken from $[0, 0.2]$.
- HBR: For HBR, we must choose the prior variances $\alpha_u, \alpha_i, \alpha_{\sigma i}, \alpha_{\sigma i}$. In every case, we sample their values from $[1e-5, 1e-1]$.

These candidate hyper-parameter values were chosen based on pilot runs. Using results from the pilot runs, we ensured that 'winning' values did not fall close to the boundaries of the range of candidate values, which would suggest the optimal value might lie outside of the range. When a 'winning' value did fall close to a boundary, we extended the range of values.

Following the same reasoning from Chapter 4, to ensure fairness in our comparisons, we set the user and item latent embeddings size d to 128 for all models. Notice that this means that both the relevance and uncertainty estimators are based on latent factors that are 128 wide.

Again, to avoid the need to tune the number of training iterations, we employ early-stopping to end the learning phase when the MAP@10 on the validation set does not improve for three consecutive iterations.

5.4.3.2 Expected correlations evaluation

Before evaluating the methods according to recommendation accuracy, for each model, we first look at the correlation between the average relevance μ_{ui} and uncertainty σ^2_{ui}. (In the case of HBR-Y, we look at the correlation between the average probability of interaction $E[\mathbb{P}_{ui}]$ and the uncertainty $\text{Var}[\mathbb{P}_{ui}]$). Having a positive correlation between the two is what motivates the additive form of ranking employed by MAR [WFZ+22]

We follow the same procedure from previous chapters. That is, we obtain the average relevance and label uncertainty from a randomly sampled set of 100,000 user-item pairs. Then, we plot the dispersion of the obtained values, together with a least squares tendency line, and analyse the results visually.

5.4.3.3 Top-n recommendation evaluation

We examine the recommendation accuracy for each model according to MAP and Recall at $n = 10$ as a function of λ. In practice, λ is a parameter that should be fixed in production. Here, we plot the results for a range of λ values to illustrate how it affects accuracy. Again, we remind the reader that ReBR is the particular case of MAR with $\lambda = 1$.

In this part of the evaluation, we consider as the candidates for recommendation to the user all items except those the user has interacted with (in the training or validation set). In the next part of the evaluation, we will restrict the candidates to long-tail items.

5.4.3.4 Long-tail recommendation evaluation

In Section 5.1.3, we argued that MAR may lead to recommendations that exhibit less popularity bias. To evaluate this, we will employ an additional evaluation protocol, introduced in [WFZ+22], which focuses on long-tail items. We will refer to this as long-tail (LT) recommendation. This protocol forces the system to recommend only tail items, and measures the accuracy of these recommendations.

For long-tail recommendation, we must define what we mean by tail items. We adopt the same method as in [WFZ+22]. That is, tail items are those with fewest interactions such that those interactions comprise 50% of all the observed interactions. We denote the set of tail items as I_{LT}. Then, for long-tail recommendation evaluation, the candidate items for recommendation to each user are given by $R_{u\cdot} \cap I_{LT}$. Similarly, the test items for each user are also confined to long-tail items: $T_{uLT} = T_u \cap I_{LT}$, where T_u are u's test set items (Section 2.5.2). Let Z^n_{uLT} denote the top-n recommended to user u based on candidates $R_{u\cdot} \cap I_{LT}$. Then, we define the MAP and Recall at n relative to the long-tail items as,

$$\text{Recall Relative@n}_u = \frac{\#\{Z^n_{uLT} \cap T_{uLT} \cap\}}{\#T_{uLT}}$$
$$\text{Precision Relative@n}_u = \frac{\#\{Z^n_{uLT} \cap T_{uLT}\}}{n}$$
$$\text{AP Relative@n}_u = \frac{1}{\#T_u} \sum_{k=1}^{n} \text{Precision Relative@k}_u \times \delta(Z^n_u(k) \in T_{uLT})$$
$$\text{MAP Relative@n} = \frac{1}{\#U} \sum_{u \in U} \text{AP Relative@n}_u$$

These are analogous to definitions in Section 2.5.2, but confined to long-tail items.

5.5 Results

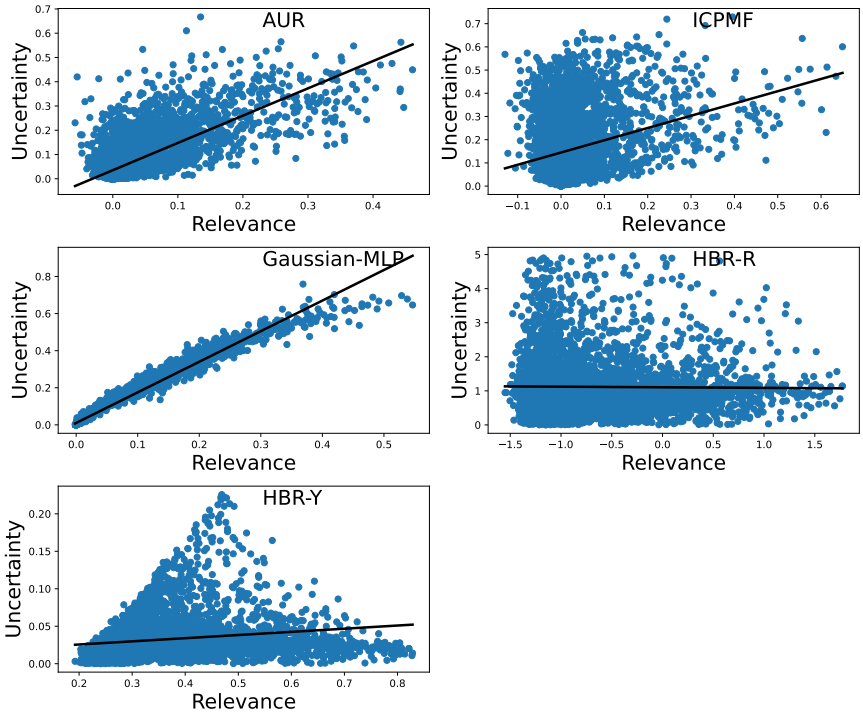
In this section, we present the results. We evaluate both general top-n recommendations (Section 5.5.2) and long-tail recommendations (Section 5.5.3). Before that, we begin with a discussion of the correlation between relevance and label uncertainty.

5.5.1 Expected correlations

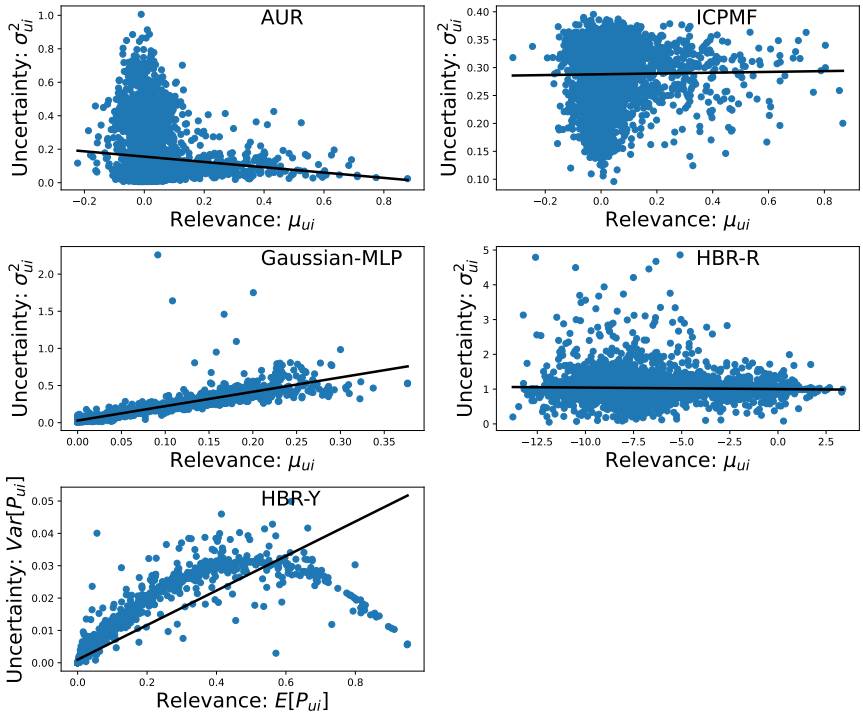
We begin by examining the correlation between the average relevance and label uncertainty. Figure 5.2 illustrates the dispersion diagrams with least squares tendency lines added.

We observed the expected correlation with AUR, ICPMF and Gaussian-MLP for the MovieLens dataset. On the other hand, the same does not hold for AUR and ICPMF for the Pinterest dataset. One case that draws attention is AUR for the Pinterest data, where there are several interactions with low relevance and high uncertainty. When ranking with MAR, such items will be more likely to be recommended for smaller values of λ.

HBR-R and HBR-Y show no apparent correlation in almost every case: the tendency lines are approximately flat for both models in the MovieLens dataset and for HBR-R on Pinterest. The exception is HBR-Y on Pinterest, where, although the tendency line still points to a positive correlation, we see that the correlation is not strictly increasing. This is perhaps to be expected. The uncertainties estimated by HBR-Y approximate 0 as $E[\mathbb{P}_{ui}]$ approximate either 0 or 1. This is



(a) MovieLens dataset.



(b) Pinterest dataset.

Figure 5.2: Dispersion diagrams between the estimated average relevances and mislabeling uncertainties for each method and dataset. The black line in each diagram is a least squares tendency line to assist visualization.

a consequence of the binary formulation. If $E[\mathbb{P}_{ui}]$ is close to 0 (or 1), then all the probability mass for $\mathbb{P}_{ui}$ is concentrated around (0 or 1), and therefore, its variance is small.

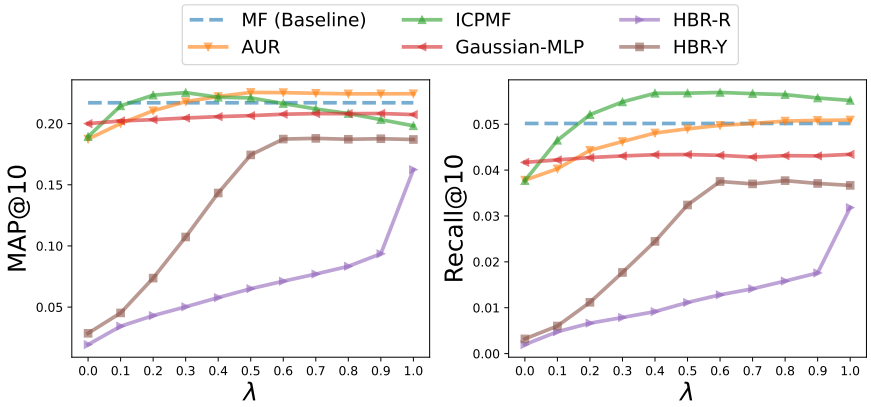
As we pointed out earlier, even though we have found that we do not always observe the anticipated correlations, this does not necessarily signal any problem with the uncertainty estimators. It shows only that using MAR with those estimators may not lead to more accurate recommendations than ReBR. Another reason why these results need not be too concerning is because we have observed (on pilot runs) that the relevance-uncertainty dispersion is affect by several hyper-parameters, especially β and γ from Eq. 5.4 and the size of the negative sample. It may be possible to have another hyper-parameter configuration that leads to models with similar accuracy but different relevance-uncertainty dispersion.

5.5.2 Top-n recommendations

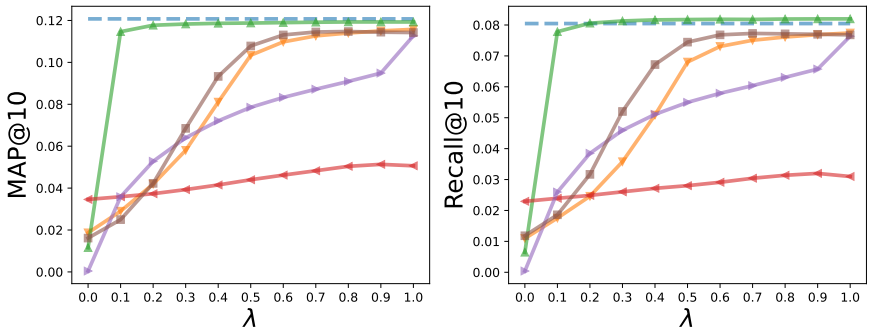
Now we examine the recommendation accuracy for each model according to MAP and Recall at $n = 10$ as a function of λ. In practice, λ is a parameter that should be fixed in production. Here, we plot the results for a range of λ values to illustrate how it affects accuracy. We note that the MF baseline accuracy, of course, has no label uncertainty estimator, and that is why its accuracy does not vary with λ.

Let's initially focus on the choice of λ. We note that for many models, the choice $\lambda = 1$ – that is, recommendations given through ReBR – are the most accurate. In fact, this is the case for all models with the Pinterest data. This is in line with what we might expect from the correlation plots. The only exceptions are ICPMF and HBR-Y on MovieLens. For the former, MAP@10 is at its maximum at $\lambda = 0.3$ and Recall@10 at $\lambda = 0.4$. For the latter, both MAP and Recall are at their maximum at $\lambda = 0.6$. Therefore, we conclude that in some specific cases HBR may be beneficial, although it is often worse than ReBR.

Now, comparing the models against each other, we note the great overall performance of ICPMF. For every metric and dataset, ICPMF is always among the most accurate systems. This is a remarkable result because ICPMF is arguably the simplest method: compared to AUR, its uncertainty estimator is of much lower dimensionality; and, compared to HBR, its learning is exact (i.e. no approximation such as that given by variational inference is required); in general, it is a lot more straightforward. Gaussian-MLP, on the other hand, achieved poor overall results. In particular, it achieved very low recommendation accuracy com-



(a) MovieLens dataset.



(b) Pinterest dataset.

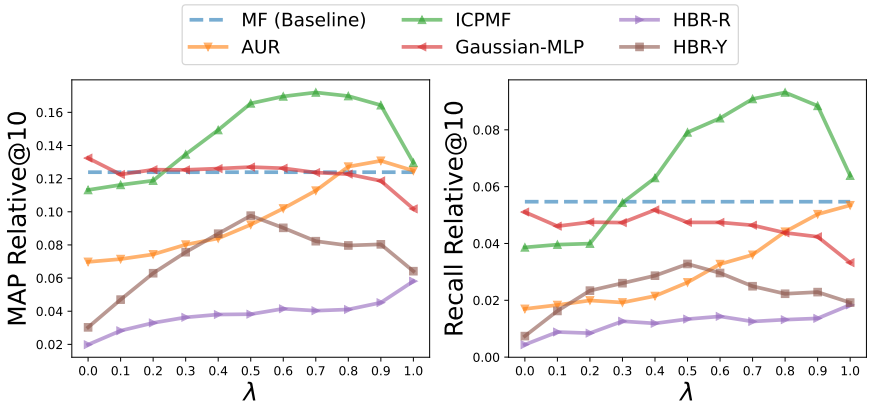
Figure 5.3: MAP@10 and Recall@10 for every model compared in the MovieLens (top) and Pinterest (bottom) datasets as a function of λ.

pared to all the other models on the Pinterest dataset. From these results, we conclude that unifying F and G, as discussed in Section 5.2.2, looks unpromising.

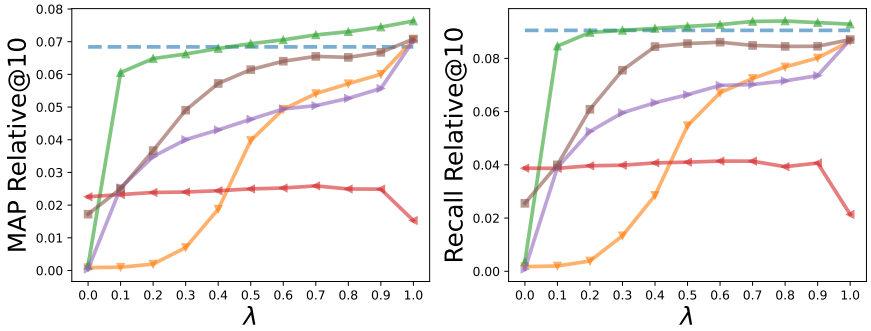
In addition, ICPMF achieved better recommendation performance than the baseline MF (MovieLens data) or almost equivalent to the baseline (Pinterest data) . Moreover, AUR is also more accurate then the baseline for MovieLens, although it is worse on Pinterest.

We also observe a difference between the results for the two datasets. On the one hand, AUR is better than HBR-R and HBR-Y in the MovieLens dataset. On the other hand, the opposite is the case for Pinterest. In particular, HBR-Y is very close to ICPMF (the top performer) in the Pinterest data according to both metrics. This may be indicative that the variation approximation needed to learn HBR performs better on bigger datasets.

Interestingly, we note that if $\lambda = 0$, then the ranking scores are defined solely by G_{ui} – that is, F_{ui} is ignored. Even in this extreme case, we see that AUR,



(a) MovieLens dataset.



(b) Pinterest dataset.

Figure 5.4: MAP Relative@10 and Recall Relative@10 for every model compared in the MovieLens (top) and Pinterest (bottom) datasets as a function of λ but for log-tail items only.

ICPMF and Gaussian-MLP are still able to recommend with decent accuracy in the MovieLens data. Therefore, another conclusion is that the label uncertainties, at least in these cases, are in fact related to relevance as we supposed.

5.5.3 Long-tail recommendations

As described in Section 5.4.3.4, we also consider the alternative setting where candidate items for recommendation are restricted to long-tail items. Figure 5.4 displays the results.

Again, let's focus initially on the choice of λ. When restricting recommendations to the long-tail items, we observe λ having a higher impact in many cases, especially in the MovieLens dataset. For this dataset we observe a major advantage of MAR over ReBR for ICPMF – in this case, MAP Relative@10 is at its highest at $\lambda = 0.7$ and Recall Relative@10 at $\lambda = 0.8$. The same is observed for HBR-Y:

both metrics are at their maximum when $\lambda = 0.5$. Interestingly, Gaussian-MLP is best when $\lambda = 0$. Unfortunately, the MAR results for the Pinterest are not as positive. In fact ReBR – the case $\lambda = 1$ – is better or very close to MAR in almost every case. The only exception is Gaussian-MLP, but its accuracy is by far the lowest. Therefore, similarly to the general top-n recommendation case, MAR was also not proven advantageous in the long-tail recommendation case for the Pinterest dataset.

Next, let's compare the models. In the MovieLens dataset, following the same trend from the overall top-n recommendation case, the performance of ICPMF is also remarkable for long-tail recommendation. We observe that ICPMF has a big advantage against every other method. Moreover, ICPMF is also a lot more accurate than the MF baseline in this dataset. For Pinterest, we found that the MAP and Recall of all methods, except Gaussian-MLP, are comparable to those of the baseline. That is, even though MAR is not beneficial in this dataset, the use of our methods with ReBR provides long-tail recommendations that are at least close in accuracy to the baseline, and, in the case of ICPMF, sometimes slightly better.

5.6 Conclusions and Limitations

In this chapter, we explored label uncertainty in IRSs. By learning to estimate this form of uncertainty, which is rather different from the prediction uncertainty that we explored in Chapters 3 and 4, we aim to counter the mislabeling problem in the learning of IRSs models, which is a consequence of negative sampling. We have introduced four novel methods for label uncertainty estimation. Our main outcomes and findings are:

- We drew a parallel between label uncertainty (and, in particular, its use in Mislabeling-Aware Ranking) and the observation model from [SYN+20]. This extends the discussion in [WFZ+22] and provides a stronger theoretical support to the exploration of label uncertainty.
- We found that ICPMF, which has a very simple label uncertainty model, is generally a good choice. In fact, this method was uniformly better than every other across all our experiments. That is, it exhibited leading accuracy in both top-n recommendations and long-tail recommendations in both the MovieLens and Pinterest datasets according to every metric. The effectiveness of ICPMF recalles the effectiveness of CPMF in Chapter 3.

- We also found that MAR may in fact be a powerful tool to combat popularity bias in some cases. For the MovieLens dataset, we showed that the use of MAR, with the correct choice of λ, can sometimes lead to major accuracy improvements. The improvement is most remarkable with ICPMF. Unfortunately, we did not observe the same for the Pinterest dataset.

Again, we acknowledge that our empirical study has some limitations, listed below. From these limitations, we may derive new research questions for exploration in future work (see Chapter 6).

- First, we have restricted ourselves to MF models as the standard choices for F and G. Although this choice allows for a fair comparison between the methods, MF is only one of many possible choices for both. We believe there is clear value in expanding the comparison to other choices. In particular, one could explore the 'interaction effect' between F and G. For instance, the impact on G given different choices of F, and vice versa.
- Second, as in the previous chapters, we considered only two datasets in our study. Although we observed a clear winner across our results (ICPMF), we also noticed that certain methods – especially HBR – performed differently in the two datasets. To explore this further, it would be useful to extend the comparison to other recommendation datasets.
- Finally, the only use we have made of label uncertainty estimates is in MAR. Although, we suppose that label uncertainty may be exploited in other ways. For example, there may be ranking criteria other than MAR, that could be tried. There may even be uses outside of top-n recommendations, such as uses in active learning.

Chapter 6

Conclusions and Future Work

This dissertation presents by far the most comprehensive study of uncertainty in Recommender Systems to date. More precisely, we distinguish between prediction uncertainty and label uncertainty, and we present several methods for estimating each of them through the various chapters. In addition, we also discussed and extended the evaluation and the use cases for uncertainty estimates in both explicit and implicit feedback-based recommender systems.

In this chapter, we present a summary of our main findings. Then, we discuss potential paths for future research in the field.

6.1 Conclusions

6.1.1 Prediction uncertainty in ERSs

In Chapter 3, we surveyed and expanded the literature in prediction uncertainty in ERSs. First, we identified several existing methods that were never compared theoretically nor empirically. We re-wrote all of these methods to use a common notation. We also proposed new methods (ENSEMBLE and EB-Linear) by adapting existing methods. Then we investigated and extended the literature on the evaluation of prediction uncertainty estimates. Finally, we implemented all the methods and compared them. All this work should greatly help further research in the field, especially because all our implementations are open source and our results are therefore reproducible.

Our empirical study showed that several of the surveyed methods are successful at prediction uncertainty estimation, especially in the rating prediction case. We

also found CPMF to be a powerful method for Uncertainty-Based Filtering and Probability-of-Relevance Ranking. Its use led to more accurate recommendations in both the datasets that we used.

6.1.2 Prediction uncertainty in IRSs

In Chapter 4, we built on the work in Chapter 3, but now turning our attention to IRSs. We noted that there was close to no literature in this case. Therefore, a major part of the chapter's contribution was the identification of the uncertainty estimation and evaluation methods from Chapter 3 that were also extensible to IRSs. In addition, we also have a strong emphasis on neural network-based RSs in this chapter. This is because (i) they are very prominent in the IRSs literature, and (ii) we could take advantage of general methods for uncertainty estimation in neural networks. Even though there are no novel methods in this chapter, their use in the problem of estimating prediction uncertainty in IRSs is novel. The chapter culminates in another extensive, and also reproducible, empirical study comparing all the explored methods.

Unfortunately, the takeaways from this chapter are not too positive. On the good side, we found that many of our explored methods (most notably, those based on ensembling) are more accurate at top-n recommendation than our baseline methods. But, different from the explicit-feedback case, we found no further benefits to using any of our uncertainty estimators in the top-n recommendations case. Most of the methods fail according to the URI metric and were not beneficial when used for Uncertainty-Based Filtering.

6.1.3 Label uncertainty in IRSs

In Chapter 5, we tackle label uncertainty in IRSs. Although label uncertainty is a widely recognized issue, it is very unexplored. In fact, the chapter builds upon the only existing method for label uncertainty estimation, Aleatoric Uncertainty-aware Recommendation (AUR). Apart from very extensively discussing the mislabeling issue, we also introduce four novel methods for label uncertainty estimation. The modeling work presented is very substantial, especially because our methods are very different from each other, including neural-networks and Bayesian hierarchical modeling. Again, we conclude the chapter with another empirical study, where the goal is to validate the use of Mislabeling-Aware Ranking (MAR) in top-n recommendations and long-tail recommendations.

Our results had two major conclusions. First, we observed that ICPMF, one of our proposed methods, is better than AUR in both the datasets that we used. This is very impressive given that ICPMF is a lot less complex than all the other methods in the comparison, including AUR. Second, we found that, in some cases, we may greatly improve a system's accuracy in long-tail recommendations by exploiting label uncertainty estimates and using MAR to select the recommendations.

6.2 Future work

In this dissertation we shed light on the quite unexplored field of uncertainty in recommender systems. Although our work is very comprehensive, we hope it will encourage further extensions – in particular, to tackle the limitations that we identified at the end of Chapters 3, 4 and 5 – and the exploration of novel research questions in the field. Below, we give an overview of what we consider to be good directions for future exploration.

6.2.1 Beyond-accuracy recommendation objectives

In Chapter 2, we briefly reviewed some of the most important beyond-accuracy recommendation objectives. In addition, we expressed our belief that uncertainty quantification might also be the key to improving beyond-accuracy recommendation goals, e.g. novelty, diversity and serendipity [KB16]. For example, a serendipitous recommendation can be one with high relevance yet high uncertainty. The subsequent chapters do not further explore these objectives. We consider this to be the most immediate and most important avenue for future research.

6.2.2 Online evaluation and user trials

In this dissertation, we restricted our experiments to offline evaluation in every case. As introduced in Chapter 2, online evaluation is an alternative that better reflects the dynamic nature of RSs. Given the extensiveness of our experiments, it was impractical to perform online evaluation. Nevertheless, there is clear value in exploring the methods present here – especially those with better performance – in an online setup.

Taking one more step forward, the evaluation of uncertainty could arguably benefit even more from user trials. To give a tangible example, we already discussed our belief that uncertainty may correlate to serendipity. The difficulties in measuring serendipity offline [KB16] impose great challenges to our ability to test

this hypothesis. For this reason, one important avenue for future research is to conduct user trials, in which users would provide feedback, not only on their item preferences, but also on beyond-accuracy questions, such as their surprise regarding recommendations that have different levels of uncertainty.

6.2.3 Long-tail recommendations

In Chapter 5, we noted the value of models that estimate label uncertainty for long-tail recommendations, to counter-act popularity bias. We even compared the models that we defined in the chapter for their long-tail performance. But the RS literature offers a plethora of other methods, for counter-acting popularity bias [LFW23, KEJ+22, BS21, AN18, WFC+21]. A line of future work would be to conduct a more extensive comparison between the methods from Chapter 5 and these other methods from the literature.

6.2.4 Label uncertainty in ERSs

In this dissertation, we have investigated label uncertainty for the implicit feedback case. Certainly, this is the most important case. But one could argue that label uncertainty is also present in explicit systems. On the one hand, we note that the problem is definitely not as big in the explicit case (where unobserved interactions are treated as missing data) as in the implicit case (where they are presumed irrelevent and can be sampled as negative interactions). Nevertheless, interactions in ERSs may be 'mislabeled' as being missing. This happens, for example, when the user consumes an item but chooses not to provide a rating. Therefore, label uncertainty could also be investigated for ERSs.

6.2.5 Novel forms of uncertainty-aware ranking

Throughout this dissertation, we discussed methods for prediction uncertainty-aware ranking, namely UBF and PRR and label uncertainty-aware ranking, namely MAR. We believe that there is a great creative space for other novel forms of uncertainty-aware ranking (for both prediction uncertainty and label uncertainty).

Some interesting ideas can be borrowed from the the related field of Information Retrieval. For example, Penha & Hauff explore uncertainty in neural learning-to-rank models [PH21]. They obtain uncertainty estimates for a BERT ranker with the use of Monte-Carlo Dropout and Deep-Ensembles, which we explained

in Chapter 4. Their uncertainty-aware ranking method combines the predicted interaction relevance with their estimated uncertainties. They found that 'shrinking' the relevance of interactions with high relevance can sometimes improve the system's recommendation accuracy. In a similar vein, but now using models based on Gaussian Processes, Guiver & Snelson proposed to either shrink or increase the relevance of items based on their uncertainty to make the model more conservative or more risk-taking [GS08].[1]

6.2.6 Other relevant comparisons

There is some work on uncertainty, or related to uncertainty, in the RSs literature that we did not cover in this dissertation. Mainly, we chose not to cover them because they tackle uncertainty from a different scope or purpose. Nevertheless, we believe that future research could try to bridge the gap between them and the research in this dissertation. We list the most relevant of them here and explain why we did not cover them.

- Bernardis et al. showed that there is a strong correlation between the eigenvalues of the item similarity matrix and the accuracy of item-based recommenders [BFDC19]. Because of this, they propose an eigenvalue confidence index to measure the confidence level of the recommendations given to each user. Their method is suitable for both explicit and implicit recommendation tasks, and confidence can be thought of as the inverse of prediction uncertainty. However, their method is applicable only to systems based on item similarity. Furthermore, like NEG-USER-SUPPORT, their confidence index is a user-centric measure, and therefore it lacks the granularity that is needed to differentiate the uncertainty of the individual items being recommended to a user.

- Another method, which is superficially related to the ones employed herein but is actually quite different in its purpose, is the use of Gaussian embeddings for collaborative filtering. Gaussian embeddings are a generalisation of the embeddings used in many recommender algorithms. In [DSPG17], Gaussian embeddings are learned by a matrix factorization algorithm; in [JYXS20], they are learned by a convolutional neural model. Gaussian embeddings give us non-deterministic user and item representations, capturing the uncertainty that there is in learning these representations. However,

[1] We note, however, that their results are largely negative: they did not find this form of ranking to be more accurate.

they do not quantify the prediction or label uncertainty of interactions.

- Neupane et al. [NZY21] propose a method for quantifying the amount of evidence available when providing recommendations to cold-start users. Without jumping into details, their method combine traditional user and item embeddings with a meta evidential learning module that is able to predict what they call evidential uncertainty. We suspect that evidential uncertainty is tightly connected to our information-based methods. Therefore, we believe that future research could formalize the similarities and differences between prediction uncertainty and evidential uncertainty, and compare our methods against theirs, where applicable.
- Another general method for uncertainty estimation are conformal predictions. The literature applying conformal predictions within ML is vast, but so far they have received received little attention in RSs. Among the few existing works, we highlight two, one by Kagitaa et. al. [KPP+17] and the other by Himabindu et. al. [HPP18]. These contain methods that could, in theory, be compared with ours. However, unlike the methods in this dissertation, their conformal predictions are not point estimates of label uncertainty. Instead, conformal predictions provide a set of possible labels (e.g. a set of points in a rating scale) for each user-item interaction. For this reason, it is not too straightforward to compare their methods against ours.

The sections above provide numerous possible paths for future research. But they are only a sample of everything that can be studied in the field. Many other directions can be suggested. For example, one could investigate the connection between label uncertainty and the definition of Missing-Not-At-Random (MNAR) data in RSs [Ste10, WZSQ19]. Or, going one step further, one could compare label uncertainty with causal [BV18] or counterfactual recommendations [WFH+21, XRK+20], since they all tackle similar implicit feedback reliability issues. The fact is that this dissertation could be the starting point for much future research.

Acronyms

ADVI Automatic Differentiation Variational Inference.

AL Active Learning.

AUR Aleatoric Uncertainty-aware Recommendation.

BBB Bayes By Back-propagation.

BCE Binary Cross-Entropy.

BNN Bayesian Neural Network.

BPR Bayesian Personalized Ranking.

CA Context-Aware.

CB Content-Based.

CF Collaborative Filtering.

CPMF Confidence-aware Probabilistic Matrix Factorization.

EB Error-Based.

ERS Explicit feedback-based Recommender System.

ERSs Explicit feedback-based Recommender Systems.

GBR Gaussian Binary Ranking.

GPR Gaussian Pairwise Ranking.

HBR Hierarchical Binary Ranking.

ICPMF Implicit Confidence-aware Probabilistic Matrix Factorization.

IRS Implicit feedback-based Recommender System.

IRSs Implicit feedback-based Recommender Systems.

MAE Mean Absolute Error.

MAP Mean Average Precision.

MAR Mislabeling-Aware Ranking.

MCDropout Monte Carlo Dropout.

MCMC Monte-Carlo Markov Chains.

MF Matrix Factorization.

ML Machine Learning.

ML1M MovieLens 1M.

MLP Multi-Layer Perceptron.

MNAR Missing-not-at-random.

MRR Mean Reciprocal Rank.

MSE Mean Squared Error.

NDCG Normalized Discounted Cumulative Gain.

NN Neural Network.

NNs Neural Networks.

PMF Probabilistic Matrix Factorization.

POP Popularity.

PRR Probability-of-Relevance Ranking.

RaBR Rating-Based Ranking.

ReBR Relevance-Based Ranking.

RMSE Root Mean Square Error.

RS Recommender System.

RSs Recommender Systems.

SVI Stochastic Variational Inference.

UAC Uncertainty Accuracy Correlation.

UBF Uncertainty-Based Filtering.

URI Uncertainty Recommendation Improvement.

VI Variational Inference.

References

[AABA22] Abdul Basit Ahanger, Syed Wajid Aalam, Muzafar Rasool Bhat, and Assif Assad. Popularity bias in recommender systems – a review. In *International Conference on Emerging Technologies in Computer Engineering*, pages 431–444. Springer, 2022.

[ABM19] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. Managing popularity bias in recommender systems with personalized re-ranking. *arXiv Preprint arXiv:1901.07555*, 2019.

[AKK07] Gediminas Adomavicius, Sreeharsha Kamireddy, and YoungOk Kwon. Towards More Confident Recommendations: Improving Recommender Systems Using Filtering Approach Based on Rating Variance. In *Proceedings of the Seventeenth Workshop on Information Technology and Systems*, pages 152–157, 2007.

[AMBM19] Himan Abdollahpouri, Masoud Mansoury, Robin Burke, and Bamshad Mobasher. The unfairness of popularity bias in recommendation. *arXiv Preprint arXiv:1907.13286*, 2019.

[AN18] Behnoush Abdollahi and Olfa Nasraoui. Transparency in fair machine learning: The case of explainable recommender systems. *Human and Machine Learning: Visible, Explainable, Trustworthy and Transparent*, pages 21–35, 2018.

[APO09] X. Amatriain, J.M. Pujol, and N. Oliver. I Like It... I Like It Not: Evaluating User Ratings Noise in Recommender Systems. In *Proceedings of the Seventeenth Conference on User Modeling, Adaptation, and Personalization*, pages 247–258, 2009.

[AS80] Jhon Atchison and Sheng M Shen. Logistic-normal distributions: Some properties and uses. *Biometrika*, 67(2):261–272, 1980.

[ASY+19] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the Twenty-fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[AT10] Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender systems. In *Recommender Systems Handbook*, pages 217–253. Springer, 2010.

[BBC20] Ricardo Mitollo Bertani, Reinaldo AC Bianchi, and Anna Helena Reali Costa. Combining novelty and popularity on personalised recommendations via user profile learning. *Expert Systems With Applications*, 146:113149, 2020.

[BBG13] Djallel Bouneffouf, Amel Bouzeghoub, and Alda Lopes Ganarski. Risk-Aware Recommender Systems. In *Proceedings of the International Conference on Neural Information Processing*, pages 57–65, 2013.

[BBK19] Edmon Begoli, Tanmoy Bhattacharya, and Dimitri Kusnezov. The need for uncertainty quantification in machine-assisted medical decision making. *Nature Machine Intelligence*, 1(1):20–23, 2019.

[BC12] Suhrid Balakrishnan and Sumit Chopra. Collaborative ranking. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, pages 143–152, 2012.

[BCJ+18] Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D. Goodman. Pyro: Deep Universal Probabilistic Programming. *Journal of Machine Learning Research*, 2018.

[BCKW15] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. In *Proceedings of the Twenty-Second International Conference on Machine Learning*, pages 1613–1622, 2015.

[BF99] Carla E Brodley and Mark A Friedl. Identifying mislabeled training data. *Journal of Artificial Intelligence Research*, 11:131–167, 1999.

[BFDC19] Cesare Bernardis, Maurizio Ferrari Dacrema, and Paolo Cremonesi. Estimating Confidence of Individual User Predictions in Item-Based Recommender Systems. In *Proceedings of the Twenty-seventh ACM Conference on User Modeling, Adaptation and Personalization*, pages 149–156, 2019.

[BGOZ18] Jesus Bobadilla, A. Gutiérrez, F. Ortega, and B. Zhu. Reliability Quality Measures for Recommender Systems. *Information Sciences*, 442:145–157, 2018.

[Bis06] Christopher M Bishop. Pattern recognition and machine learning. *Springer Google Scholar*, 2:645–678, 2006.

[BL07] James Bennett and Stan Lanning. The Netflix Prize. In *Proceedings of KDD Cup and Workshop*, pages 3–6, 2007.

[Blo98] David Block. Exploring interpretations of questionnaire items. *System*, 26(3):403–425, 1998.

[BMNG18] Mark Belford, Brian Mac Namee, and Derek Greene. Stability of Topic Modeling via Matrix Factorization. *Expert Systems With Applications*, 91:159–169, 2018.

[BRL06] Christopher Burges, Robert Ragno, and Quoc Le. Learning to rank with nonsmooth cost functions. *Advances in Neural Information Processing Systems*, 19, 2006.

[BS21] Rodrigo Borges and Kostas Stefanidis. On mitigating popularity bias in recommendations via variational autoencoders. In *Proceedings of the Thirty-sixth annual ACM Symposium on Applied Computing*, pages 1383–1389, 2021.

[BSDN+18] Vito Bellini, Angelo Schiavone, Tommaso Di Noia, Azzurra Ragone, and Eugenio Di Sciascio. Knowledge-aware autoencoders for explainable recommender systems. In *Proceedings of the Third Workshop on Deep Learning for Recommender Systems*, pages 24–31, 2018.

[BU17] Andrea Barraza-Urbina. The Exploration-Exploitation Trade-Off in Interactive Recommender Systems. In *Proceedings of the 11th ACM Conference on Recommender Systems*, pages 431–435, 2017.

[Bur02a] Robin Burke. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, 12(4):331—-370, 2002.

[Bur02b] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12:331–370, 2002.

[Bur10] Christopher JC Burges. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81, 2010.

[BV18] Stephen Bonner and Flavian Vasile. Causal embeddings for recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 104–112, 2018.

[CAH22] Matthew J Colbrook, Vegard Antun, and Anders C Hansen. The difficulty of computing stable and accurate neural networks: On the barriers of deep learning and smale's 18th problem. *Proceedings of the National Academy of Sciences*, 119(12):e2107151119, 2022.

[Car20] Diego Carraro. *Active Learning in Recommender Systems: An Unbiased and Beyond-Accuracy Perspective*. PhD thesis, University College Cork, 2020.

[CB22] Victor Coscrato and Derek Bridge. Recommendation uncertainty in implicit feedback recommender systems. In *Proceedings of the 30th Irish Conference on Artificial Intelligence and Cognitive Science*. Springer, 2022.

[CHV21] Pablo Castells, Neil Hurley, and Saul Vargas. Novelty and diversity in recommender systems. In *Recommender Systems Handbook*, pages 603–646. Springer, 2021.

[CKH+16] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the First Workshop on Deep Learning for Recommender Systems*, pages 7–10, 2016.

[CLA+03] Dan Cosley, Shyong K Lam, Istvan Albert, Joseph A Konstan, and John Riedl. Is seeing believing? how recommender system interfaces affect users' opinions. In *Proceedings of the SIGCHI*

Conference on Human Factors in Computing Systems, pages 585–592, 2003.

[CSE18] Allison J. B. Chaney, Brandon M. Stewart, and Barbara E. Engelhardt. How Algorithmic Confounding in Recommendation Systems Increases Homogeneity and Decreases Utility. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 224–232, 2018.

[CTFLHT13] Sergio Cleger-Tamayo, Juan M Fernández-Luna, Juan F Huete, and Nava Tintarev. Being confident about the quality of the predictions in recommender systems. In *European Conference on Information Retrieval*, pages 411–422. Springer, 2013.

[CYW+19] Li Chen, Yonghua Yang, Ningxia Wang, Keping Yang, and Quan Yuan. How serendipity improves user satisfaction with recommendations? a large-scale user evaluation. In *The World Wide Web Conference*, pages 240–250, 2019.

[Dau17] Jean Daunizeau. Semi-analytical approximations to statistical moments of sigmoid and softmax mappings of normal variables. *arXiv Preprint arXiv:1703.00091*, 2017.

[Daw82] A Philip Dawid. The well-calibrated bayesian. *Journal of the American Statistical Association*, 77(379):605–610, 1982.

[dCMC18] Arthur F. da Costa, Marcelo G. Manzato, and Ricardo J. G. B. Campello. CoRec: A Co-Training Approach for Recommender Systems. In *Proceedings of the Thirty-third Annual ACM Symposium on Applied Computing*, page 696–703, 2018.

[DGBC22] Edoardo D'Amico, Giovanni Gabbolini, Cesare Bernardis, and Paolo Cremonesi. Analyzing and improving stability of matrix factorization for recommender systems. In *Journal of Intelligent Information Systems*, 2022.

[DMRAB+19] Jorge Díez, David Martínez-Rego, Amparo Alonso-Betanzos, Oscar Luaces, and Antonio Bahamonde. Optimizing novelty and diversity in recommendations. *Progress in Artificial Intelligence*, 8:101–109, 2019.

[DSPG17] Ludovic Dos Santos, Benjamin Piwowarski, and Patrick Gallinari. Gaussian embeddings for collaborative filtering. In *Proceedings of*

the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 1065–1068, 2017.

[FL08] Patrizio Frederic and Frank Lad. Two moments of the logitnormal distribution. *Communications in Statistics—Simulation and Computation extregistered*, 37(7):1263–1269, 2008.

[Fun06] Simon Funk. Netflix Update: Try This at Home. `https://sifter.org/simon/journal/20061211.html`, 2006.

[FX22] Shuyi Fang and David Jingjun Xu. Alleviating information cocoons and fatigue with serendipity: Effect of relevant diversification and its timing. In *Proceedings of the Forty-third International Conference on Information Systems*, 2022.

[GCSR95] Andrew Gelman, John B Carlin, Hal S Stern, and Donald B Rubin. *Bayesian Data Analysis*. Chapman and Hall/CRC, 1995.

[GDBJ10] Mouzhi Ge, Carla Delgado-Battenfeld, and Dietmar Jannach. Beyond accuracy: Evaluating recommender systems by coverage and serendipity. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, pages 257–260, 2010.

[Gey92] Charles J Geyer. Practical Markov Chain Monte Carlo. *Statistical Science*, pages 473–483, 1992.

[GF20] Ethan Goan and Clinton Fookes. Bayesian Neural Networks: An Introduction and Survey. In *Case Studies in Applied Bayesian Data Science*, pages 45–87. Springer, 2020.

[GG16] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proceedings of the Thirty-third International Conference on Machine Learning*, pages 1050–1059, 2016.

[GPB10] Mustansar Ali Ghazanfar and Adam Prugel-Bennett. A scalable, accurate hybrid recommender system. In *2010 Third International Conference on Knowledge Discovery and Data Mining*, pages 94–98. IEEE, 2010.

[Gra11] Alex Graves. Practical variational inference for neural networks. In *Proceedings of the Twenty-fourth International Conference on Neural Information Processing Systems*, pages 2348—2356, 2011.

[GS08] John Guiver and Edward Snelson. Learning to Rank With SoftRank and Gaussian Processes. In *Proceedings of the Thirty-first International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 259–266, 2008.

[GTY+17] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: A factorization-machine based neural network for ctr prediction. *arXiv Preprint arXiv:1703.04247*, 2017.

[GZBC15] Xue Geng, Hanwang Zhang, Jingwen Bian, and Tat-Seng Chua. Learning image and user features for recommendation in social networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4274–4282, 2015.

[GZQ+20] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering*, 34(8):3549–3568, 2020.

[HBWP13] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 2013.

[HDW+20] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the Forty-third International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 639–648, 2020.

[HK15] F Maxwell Harper and Joseph A Konstan. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems*, 5(4):1–19, 2015.

[HKTR04] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.

[HKV08] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 8h IEEE International Conference on Data Mining*, pages 263–272, 2008.

[HLZ+17] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the Twenty-sixth International Conference on the World Wide Web*, pages 173–182, 2017.

[HPP18] Tadiparthi VR Himabindu, Vineet Padmanabhan, and Arun K Pujari. Conformal matrix factorization based recommender system. *Information Sciences*, 467:685–707, 2018.

[HS22] John B Holmes and Matthew R Schofield. Moments of the logit-normal distribution. *Communications in Statistics – Theory and Methods*, 51(3):610–623, 2022.

[HV07] Barbara Hammer and Thomas Villmann. How to process uncertainty in machine learning? In *European Symposium on Artificial Neural Networks*, pages 79–90, 2007.

[HW21] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110(3):457–506, 2021.

[JYXS20] Junyang Jiang, Deqing Yang, Yanghua Xiao, and Chenlu Shen. Convolutional gaussian embeddings for personalized recommendation with uncertainty. *arXiv Preprint arXiv:2006.10932*, 2020.

[KB16] Marius Kaminskas and Derek Bridge. Diversity, Serendipity, Novelty, and Coverage: A Survey and Empirical Analysis of Beyond-Accuracy Objectives in Recommender Systems. *ACM Trans. Interact. Intell. Syst.*, 7(1), 2016.

[KBV09] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, pages 30–37, 2009.

[KEJ+22] Anastasiia Klimashevskaia, Mehdi Elahi, Dietmar Jannach, Christoph Trattner, and Lars Skjærven. Mitigating popularity bias in recommendation: Potential and limits of calibration approaches. In *International Workshop on Algorithmic Bias in Search and Recommendation*, pages 82–90. Springer, 2022.

[KKT+15] Komal Kapoor, Vikas Kumar, Loren Terveen, Joseph A Konstan, and Paul Schrater. 'I like to explore sometimes' adapting to dy-

namic user novelty preferences. In *Proceedings of the Nineth ACM Conference on Recommender Systems*, pages 19–26, 2015.

[KM01] Arnd Kohrs and Bernard Mérialdo. Improving collaborative filtering for new-users by smart object selection. In *Proceedings of the International Conference on Media Futures (ICME)*, 2001.

[Kor08] Yehuda Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the Fourteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 426–434, 2008.

[KPP+17] Venkateswara Rao Kagita, Arun K Pujari, Vineet Padmanabhan, Sandeep Kumar Sahu, and Vikas Kumar. Conformal recommender system. *Information Sciences*, 405:157–174, 2017.

[KS11] Yehuda Koren and Joe Sill. OrdRec: An Ordinal Model for Predicting Personalized Item Rating Distributions. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, pages 117–124, 2011.

[KTR+17] Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David M Blei. Automatic differentiation variational inference. *Journal of Machine Learning Research*, 18(14):1–45, 2017.

[LCMB16] Dawen Liang, Laurent Charlin, James McInerney, and David M Blei. Modeling user exposure in recommendation. In *Proceedings of the Twenty-fifth International Conference on World Wide Web*, pages 951–961, 2016.

[LDGS11] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. *Recommender Systems Handbook*, pages 73–105, 2011.

[LFW23] Zhongzhou Liu, Yuan Fang, and Min Wu. Mitigating popularity bias for users and items with fairness-centric adaptive recommendation. *ACM Transactions on Information Systems*, 41(3):1–27, 2023.

[LJ14] Lukas Lerche and Dietmar Jannach. Using graded implicit feedback for bayesian personalized ranking. In *Proceedings of the*

Eighth ACM Conference on Recommender Systems, pages 353–356, 2014.

[LKH14] Blerina Lika, Kostas Kolomvatsos, and Stathes Hadjiefthymiades. Facing the cold start problem in recommender systems. *Expert Systems With Applications*, 41(4):2065–2073, 2014.

[LKHJ18] 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*, pages 689–698, 2018.

[LPB17] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proceedings of the Thirty-first International Conference in Neural Information Processing Systems*, pages 6405–6416, 2017.

[LRS+17] David C Liu, Stephanie Rogers, Raymond Shiau, Dmitry Kislyuk, Kevin C Ma, Zhigang Zhong, Jenny Liu, and Yushi Jing. Related pins at pinterest: The evolution of a real-world recommender system. In *Proceedings of the Twenty-sixth International Conference on World Wide Web Companion*, pages 583–592, 2017.

[Mar03] Benjamin M Marlin. Modeling User Rating Profiles for Collaborative Filtering. *Proceedings of the Sixteenth International Conference on Neural Information Processing Systems*, pages 627–634, 2003.

[Maz13] Maciej A. Mazurowski. Estimating Confidence of Individual Rating Predictions in Collaborative Filtering Recommender Systems. *Expert Systems With Applications*, 40(10):3847–3857, 2013.

[MLG+03] Sean M. McNee, Shyong K. Lam, Catherine Guetzlaff, Joseph A. Konstan, and John Riedl. Confidence Displays and Training in Recommender Systems. In *Proceedings of IFIP INTERACT03: Human-Computer Interaction*, volume 3, pages 176–183, 2003.

[MMSJS12] Joao Mendes-Moreira, Carlos Soares, Alípio Mário Jorge, and Jorge Freire De Sousa. Ensemble approaches for regression: A survey. *ACM Computing Surveys (Csur)*, 45(1):1–40, 2012.

[MRPC+21] Alessandro B Melchiorre, Navid Rekabsaz, Emilia Parada-Cabaleiro, Stefan Brandl, Oleg Lesota, and Markus Schedl. Investigating gender fairness of recommendation algorithms in the music domain. *Information Processing & Management*, 58(5):102666, 2021.

[MS08] Andriy Mnih and Russ R. Salakhutdinov. Probabilistic Matrix Factorization. In *Advances in Neural Information Processing Systems*, pages 1257–1264, 2008.

[MZLK11] Hao Ma, Tom Chao Zhou, Michael R Lyu, and Irwin King. Improving recommender systems by incorporating social contextual information. *ACM Transactions on Information Systems (TOIS)*, 29(2):1–23, 2011.

[Ngu19] Vu Nguyen. Bayesian optimization for accelerating hyperparameter tuning. In *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, pages 302–305. IEEE, 2019.

[NMS+19] Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G Azzolini, et al. Deep learning recommendation model for personalization and recommendation systems. *arXiv Preprint arXiv:1906.00091*, 2019.

[NZY21] Krishna Prasad Neupane, Ervine Zheng, and Qi Yu. Metaedl: Meta evidential learning for uncertainty-aware cold-start recommendations. In *2021 IEEE International Conference on Data Mining (ICDM)*, pages 1258–1263. IEEE, 2021.

[OHS06] Michael P. O'Mahony, Neil J. Hurley, and Guénolé CM Silvestre. Detecting noise in recommender system databases. In *Proceedings of the 11th International Conference on Intelligent User Interfaces*, pages 109–115, 2006.

[OLCGPB21] Fernando Ortega, Raúl Lara-Cabrera, Ángel González-Prieto, and Jesús Bobadilla. Providing Reliability in Recommender Systems Through Bernoulli Matrix Factorization. *Information Sciences*, 553:110–128, 2021.

[P+99] John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, 10(3):61–74, 1999.

[PGC+17] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic Differentiation in PyTorch. In *Proceedings of the NIPS 2017 Workshop on Autodiff*, 2017.

[PH21] Gustavo Penha and Claudia Hauff. On the Calibration and Uncertainty of Neural Learning to Rank Models. *arXiv:2101.04356*, 2021.

[POMT+20] Francisco J Peña, Diarmuid O'Reilly-Morgan, Elias Z Tragos, Neil Hurley, Erika Duriakova, Barry Smyth, and Aonghus Lawlor. Combining Rating and Review Data by Initializing Latent Factor Models With Topic Models for Top-N Recommendation. In *Proceedings of the Fourteenth ACM Conference on Recommender Systems*, pages 438–443, 2020.

[RFGST09] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: Bayesian Personalized Ranking From Implicit Feedback. In *Proceedings of the Twenty-fifth Conference on Uncertainty in Artificial Intelligence*, pages 452–461, 2009.

[RKZA20] Steffen Rendle, Walid Krichene, Li Zhang, and John Anderson. Neural collaborative filtering vs. matrix factorization revisited. In *Proceedings of the Fourteenth ACM Conference on Recommender Systems*, pages 240–248, 2020.

[RRS11] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to the Recommender Systems Handbook. In *Recommender Systems Handbook*, pages 1–35. Springer, 2011.

[RZS16] Ye Ren, Le Zhang, and Ponnuthurai N Suganthan. Ensemble classification and regression-recent developments, applications and future directions. *IEEE Computational Intelligence Magazine*, 11(1):41–53, 2016.

[SAA14] Boulkrinat Samia, Hadjali Allel, and Aissani Mokhtari Aicha. Handling preferences under uncertainty in recommender systems.

In *2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 2262–2269. IEEE, 2014.

[SD21] Cecilia Summers and Michael J Dinneen. Nondeterminism and instability in neural network optimization. In *International Conference on Machine Learning*, pages 9913–9922. PMLR, 2021.

[SG11] Guy Shani and Asela Gunawardana. Evaluating Recommendation Systems. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, pages 257–297. Springer, 2011.

[SLH10] Yue Shi, Martha Larson, and Alan Hanjalic. List-wise learning to rank with matrix factorization for collaborative filtering. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, pages 269–272, 2010.

[SM08] Ruslan Salakhutdinov and Andriy Mnih. Bayesian Probabilistic Matrix Factorization Using Markov Chain Monte Carlo. In *Proceedings of the Twenty-fifth International Conference on Machine Learning*, pages 880–887, 2008.

[SS13] Laila Safoury and Akram Salah. Exploiting user demographic attributes for solving cold-start problem in recommender system. *Lecture Notes on Software Engineering*, 1(3):303–307, 2013.

[Ste10] Harald Steck. Training and testing of recommender systems on data missing not at random. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 713–722, 2010.

[SYN+20] Yuta Saito, Suguru Yaginuma, Yuta Nishino, Hayato Sakata, and Kazuhide Nakata. Unbiased recommender learning from missing-not-at-random implicit feedback. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 501–509, 2020.

[TM10] Nava Tintarev and Judith Masthoff. Designing and evaluating explanations for recommender systems. In *Recommender Systems Handbook*, pages 479–510. Springer, 2010.

[VC11] Saúl Vargas and Pablo Castells. Rank and relevance in novelty and diversity metrics for recommender systems. In *Proceedings of the*

Fifth ACM Conference on Recommender Systems, pages 109–116, 2011.

[VGS05] Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*, volume 29. Springer, 2005.

[VMO+12] Katrien Verbert, Nikos Manouselis, Xavier Ochoa, Martin Wolpers, Hendrik Drachsler, Ivana Bosnic, and Erik Duval. Context-aware recommender systems for learning: A survey and future challenges. *IEEE Transactions on Learning Technologies*, 5(4):318–335, 2012.

[WFC+21] Tianxin Wei, Fuli Feng, Jiawei Chen, Ziwei Wu, Jinfeng Yi, and Xiangnan He. Model-agnostic counterfactual reasoning for eliminating popularity bias in recommender system. In *Proceedings of the Twenty-seventh ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1791–1800, 2021.

[WFH+21] Wenjie Wang, Fuli Feng, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua. Clicks can be cheating: Counterfactual recommendation for mitigating clickbait issue. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1288–1297, 2021.

[WFZ+22] Chenxu Wang, Fuli Feng, Yang Zhang, Qifan Wang, Xunhan Hu, and Xiangnan He. Rethinking missing data: Aleatoric uncertainty-aware recommendation. *arXiv Preprint arXiv:2209.11679*, 2022.

[WJ23] Lequn Wang and Thorsten Joachims. Uncertainty quantification for fairness in two-stage recommender systems. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pages 940–948, 2023.

[WLW+18] Chao Wang, Qi Liu, Runze Wu, Enhong Chen, Chuanren Liu, Xunpeng Huang, and Zhenya Huang. Confidence-Aware Matrix Factorization for Recommender Systems. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 434–442, 2018.

[WLY+19] Bin Wang, Jie Lu, Zheng Yan, Huaishao Luo, Tianrui Li, Yu Zheng, and Guangquan Zhang. Deep uncertainty quantifi-

cation: A machine learning approach for weather forecasting. In *Proceedings of the Twenty-fifth ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2087–2095, 2019.

[WMZ+23] Yifan Wang, Weizhi Ma, Min Zhang, Yiqun Liu, and Shaoping Ma. A survey on the fairness of recommender systems. *ACM Transactions on Information Systems*, 41(3):1–43, 2023.

[WWSY13] Jianan Wu, Yinglu Wu, Jie Sun, and Zhilin Yang. User reviews and uncertainty assessment: A two stage model of consumers' willingness-to-pay in online markets. *Decision Support Systems*, 55(1):175–185, 2013.

[WZSQ19] Xiaojie Wang, Rui Zhang, Yu Sun, and Jianzhong Qi. Doubly robust joint learning for recommendation on data missing not at random. In *International Conference on Machine Learning*, pages 6638–6647. PMLR, 2019.

[XRK+20] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Adversarial counterfactual learning and evaluation for recommender system. *Advances in Neural Information Processing Systems*, 33:13515–13526, 2020.

[YB21] Yinchong Yang and Florian Buettner. Multi-output gaussian processes for uncertainty-aware recommender systems. In *Uncertainty in Artificial Intelligence*, pages 1505–1514. PMLR, 2021.

[ZC+20] Yongfeng Zhang, Xu Chen, et al. Explainable recommendation: A survey and new perspectives. *Foundations and Trends extregistered in Information Retrieval*, 14(1):1–101, 2020.

[Zha13] Liang Zhang. The definition of novelty in recommendation system. *Journal of Engineering Science & Technology Review*, 6(3), 2013.

[ZHAK18] Qian Zhao, F Maxwell Harper, Gediminas Adomavicius, and Joseph A Konstan. Explicit or implicit feedback? engagement or satisfaction? a field experiment on machine-learning-based recommender systems. In *Proceedings of the Thirty-third Annual ACM Symposium on Applied Computing*, pages 1331–1340, 2018.

[ZN09] Azene Zenebe and Anthony F. Norcio. Representation, Similarity Measures and Aggregation Methods Using Fuzzy Sets for Content-

Based Recommender Systems. *Fuzzy Sets and Systems*, 160(1):76–94, 2009.

[ZOBG18] Bo Zhu, Fernando Ortega, Jesús Bobadilla, and Abraham Gutiérrez. Assigning reliability values to recommendations using matrix factorization. *Journal of Computational Science*, 26:165–177, 2018.

[ZTS+17] Yoel Zeldes, Stavros Theodorakis, Efrat Solodnik, Aviv Rotman, Gil Chamiel, and Dan Friedman. Deep density networks and uncertainty in recommender systems. *arXiv Preprint arXiv:1711.02487*, 2017.

[ZTZX14] Mi Zhang, Jie Tang, Xuchen Zhang, and Xiangyang Xue. Addressing Cold Start in Recommender Systems: A Semi-Supervised Co-Training Algorithm. In *Proceedings of the Thirty-seventh International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 73–82, 2014.