Recommendation Uncertainty in Implicit Feedback Recommender Systems

Victor Coscrato Derek Bridge

School of Computer Science & Information Technology,
University College Cork, Ireland

Abstract. A Recommender System's recommendations will each carry a certain level of uncertainty. The quantification of this uncertainty can be useful in a variety of ways. Estimates of uncertainty might be used externally; for example, showing them to the user to increase user trust in the abilities of the system. They may also be used internally; for example, deciding the balance of 'safe' and less safe recommendations. In this work, we explore several methods for estimating uncertainty. The novelty comes from proposing methods that work in the implicit feedback setting. We use experiments on two datasets to compare a number of recommendation algorithms that are modified to perform uncertainty estimation. In our experiments, we show that some of these modified algorithms are less accurate than their unmodified counterparts, but others are actually more accurate. We also show which of these methods are best at enabling the recommender to be 'aware' of which of its recommendations are likely to be correct and which are likely to be wrong.

Keywords: Recommender Systems · Uncertainty · Neural Networks

1 Introduction

Recommender Systems (RS) help users discover items in item catalogs. In general, for each user, the majority of the items are not relevant; therefore, the system's task is to select and present personalized recommendation lists to each individual user. In most RS domains, the item catalog is huge, meaning that users will have interacted with only a tiny fraction of the items. This is known as *sparsity*. It means the system must infer the user's preferences from a relatively small amount of information, leading sometimes to the generation of unsuccessful recommendations. Other factors that make it difficult to make good recommendations include changes in user mood and changes in user preferences over time. Due to the challenges of modelling ever-changing preferences from sparse feedback, there can be high uncertainty in the recommendations that an RS makes to its users [22].

It is important for an RS to quantify its uncertainty. Estimates of recommendation uncertainty can help the RS detect which of its recommendations are more or less likely to be incorrect [16]. It can use these estimates in a variety of ways. The

This work was conducted with the financial support of the Science Foundation Ireland Centre for Research Training in Artificial Intelligence under Grant No. 18/CRT/6223.

simplest is to expose them to the user: a recommendation can be accompanied by a number, a text or a visual that shows how sure or unsure the RS is that the user will like the recommendation. If a poor recommendation (one that the user does not like) is accompanied by a declaration of high uncertainty, for example, user trust may not be lost in the way that it might have been if there had been no declaration of uncertainty. As well as exposing uncertainty estimates to users, an RS may use the uncertainty estimates internally. For example, it may use them to decide how many uncertain recommendations to show (perhaps none when building trust, perhaps more when aiming for serendipity); or a hybrid RS may use them to decide when to call on different recommendation algorithms.

The literature on RS uncertainty quantification focuses almost exclusively on the rating prediction task, e.g. [22,31]. This is the case where the system has explicit feedback, usually in the form of numeric ratings (e.g. 1–5 stars) and where the task of the RS is to predict the rating for an unseen user-item pair. Of much more importance to the construction of usable RS is the top-K recommendation task and even more so in the case of implicit feedback, where the RS knows only which items a user has interacted with (e.g. which she has purchased or which she has clicked on). Yet there is almost no published research on uncertainty estimation in this more important setting. Not only that, but much of the work done on uncertainty estimation in the explicit rating prediction setting does not generalise to the implicit feedback setting. For example, in some explicit rating approaches, the ratings are assumed to follow a statistical distribution, whose dispersion is used to estimate uncertainty, e.g. [29]. This cannot generalise to the case where there there are only implicit signals, with no rating values.

However, in other fields, dominated by neural models, such as computer vision [19] and natural language processing [30], methods for estimating the network's predictive uncertainty have been explored. Among the most popular methods used for uncertainty quantification in neural networks are Bayesian Neural Networks (BNNs) [3], Monte-Carlo Dropout (MCDropout) [7] and Deep-Ensembles [20].

We see an opportunity to adapt ideas from these other fields to the field of RS. This adaptation is made easier by the fact that the state-of-the-art in recommendation algorithms has changed in recent years. Now, several neural network-based recommenders have been proposed [4,15,23], with some achieving impressive results. Several properties of neural networks explain their increasing usage. First, the flexibility of neural architectures allows for a wide variety of data inputs, making it relatively straightforward to combine interaction data with item content data, user data and contextual data [13]. Second, neural networks are a great tool for latent representation learning, as shown by the success of variational autoencoder recommenders [21]. Up to now, uncertainty estimation in these neural RS remains unexplored.

In this paper, we introduce uncertainty estimation to implicit feedback RS. More specifically, we re-purpose several uncertainty estimation methods that were successful in other tasks —either on explicit feedback recommenders or on neural networks in other fields— to make them suitable for the implicit recommender case. We compare these uncertainty estimation methods against each other, aiming to provide initial answers to two questions:

Q1: When implicit feedback RS are modified to perform uncertainty estimation, are there changes (gains or losses) in the accuracy of the RS?

Q2: Do the uncertainty estimates help the RS understand which of its recommendation are more likely to be right and which are more likely to be wrong?

The following Section (2) formulates the uncertainty estimation problem for implicit feedback RS. Section 3 proposes some techniques for solving this problem. Experiments are described in Section 4 along with their results. In Section 5, we review some related work that can be relevant to further work on RS uncertainty. Finally, Section 6 contains our conclusions.

2 Background

At its core, recommending is the task of selecting from many items those that are most relevant to the user. In this work, we focus on implicit feedback systems. In this case, the task of recommending can be seen as comprising at least two sub-tasks: first, estimating relevance scores for unobserved user-item interactions; and second, selecting the top-K items for a given user, guided by these relevance scores.

Formally, let $\mathcal{D} = \{(u,i)|u \in U, i \in I\}$ denote all the user-item pairs, where U and I are the set of users and items in the system, respectively. Users interact with items, e.g. purchasing them, clicking on them, and so on. We denote the set of observed user-item interactions by $\mathcal{D}^+ \subset \mathcal{D}$. Similarly, we denote the set of unobserved interactions $\mathcal{D}^- = \mathcal{D} - \mathcal{D}^+$. Then, for relevance scoring, the RS must learn a model $F_\theta(u,i) = r_{ui}$, parameterized by θ, from the observed interactions to predict the relevance of unseen user-item pairs.

With this setup, 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 [17]. In this work, we employ cross-entropy loss for every recommender. Nevertheless, we remark that ranking-based losses, such as Bayesian Personalized Ranking [27], could also be used instead, without affecting the uncertainty estimation methods that we will explore in Section 3.

The methods for uncertainty estimation herein can be applied to a wide class of recommender algorithms, that is, there are several possible choices for F_θ. Where possible, we will employ the well-known matrix factorization (MF) algorithm [17]. MF consists in learning a D-dimensional latent vector for each user and item. To predict r_{ui}, the user and item embeddings are combined, as follows,

$$F_\theta(u,i) = p_u^t q_i \tag{1}$$

where p_u and q_i are the user and item latent representations, respectively. In this case, $\theta = \{\{p_u\}_{u \in U}, \{q_i\}_{i \in I}\}$.

Furthermore, as explained in Section 1, we also want to use some uncertainty estimation methods that apply to neural models. For our neural recommender, we will

use one of the simplest and most popular algorithms: He et al.'s Multi-Layer Perceptron (MLP) recommender [15]. In this case,

$$F_\theta(u,i) = \text{MLP}(p_u || q_i) \tag{2}$$

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

$$f_0 = p_u || q_i \tag{3}$$

$$f_l = \text{ReLU}(W_l f_{l-1}), \quad \text{for } l \in 1, \ldots, L-1; \tag{4}$$

$$f_L = \text{Sigmoid}(w_L^t f_{L-1}) \tag{5}$$

where W_l is the weight matrix for hidden layer l and w_L is the output layer's weight vector. The Sigmoid activation in the output layer scales the output to $[0,1]$.

For both the MF and the MLP, the parameters are learned by mini-batch gradient descent, minimizing the binary cross-entropy loss. On each training epoch, the training data consists of the observed interactions D^+ and an N-sized randomly-selected sample of non-observed interactions from D^-, where N is a hyperparameter.

We now turn our attention to uncertainty estimation methods. We use σ_{ui} to denote the uncertainty associated with the predicted relevance r_{ui}.

3 Uncertainty Estimation Methods

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. In this section, we present several methods for uncertainty estimation, making clear which recommender algorithms they can be used with.

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 [22]. Furthermore, these estimates can be used with any recommendation algorithm. In the past, they have been used for explicit feedback recommenders, but here we use them in the implicit setting. These estimates can be user-centric or item-centric. Hence, following [22], we define the following uncertainty metrics,

$$\text{NEG-USER-SUPPORT} : \sigma_{ui} = -\#u \tag{6}$$

$$\text{NEG-ITEM-SUPPORT} : \sigma_{ui} = -\#i \tag{7}$$

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 r_{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.

Beyond 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 [5,25].

In the case of explicit feedback, ensembles have been successful at estimating the uncertainty of MF rating predictions [22]. 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 (Eq. 1) and also any neural network model, and in particular the MLP model (Eq. 2).[1]

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:

$$r_{ui} = \frac{\sum_{k=1}^{n} F_\theta^{(k)}(u,i)}{n} \tag{8}$$

$$\sigma_{ui} = \frac{\sum_{k=1}^{n} \left(F_\theta^{(k)}(u,i) - r_{ui}\right)^2}{n} \tag{9}$$

Bayesian Neural Networks (BNN) are another major tool tailored to uncertainty quantification in neural models. BNNs differ from their deterministic counterpart by treating the parameters as random variables [10], 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,

$$p(\theta|\mathcal{D}) = \frac{p(\theta)p(D|\theta)}{p(D)} \tag{10}$$

Calculating the posterior directly from Eq. 10 is generally not possible, because the data evidence, $p(D)$, is unknown. For this reason, inference methods such as Monte-Carlo Markov Chains (MCMC) [9] and Variational Inference (VI) [11] are applied to approximate the exact posterior. More recently, Bayes By Back-propagation (BBB) has been proposed [3], a method that allows for the posterior weights distribution to be learned through back-propagation, 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 r_{ui}, and its variance $Var[F_\theta(u,i)]$ is an estimate of relevance uncertainty σ_{ui}. In practice, the values are estimated using samples $\theta_k,\ldots,\theta_k$ from the posterior, as follows,

[1] An ensemble of neural models is often referred to as a Deep-Ensemble [20].

Table 1. Methods we compare.

Name	Prediction model (F)	Uncertainty estimator
MF-NUS	MF (Eq. 1)	NEG-USER-SUPPORT (Eq. 6)
MF-NIS	MF (Eq. 1)	NEG-ITEM-SUPPORT (Eq. 7)
MF-Ensemble	MF (Eq. 1)	Ensemble (Eqs. 8 – 9)
MLP-NUS	MLP (Eq. 2)	NEG-USER-SUPPORT (Eq. 6)
MLP-NIS	MLP (Eq. 2)	NEG-ITEM-SUPPORT (Eq. 7)
BayesianMLP	MLP (Eq. 2)	Bayesian inference (BBB) (Eqs. 11 – 12)
MCDropout	MLP (Eq. 2)	Monte-Carlo Dropout (Eqs. 8 – 9)
MLP-Ensemble	MLP (Eq. 2)	Ensemble (Eqs. 8 - 9)

$$r_{ui} = \frac{\sum_{k=1}^{n} F_{\theta_k}(u,i)}{n} \tag{11}$$

$$\sigma_{ui} = \frac{\sum_{k=1}^{n} \left(F_{\theta_k}(u,i) - r_{ui}\right)^2}{n} \tag{12}$$

Another uncertainty estimation method that is tailored to neural networks is MC-Dropout [7]. 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.[2] 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 8 and 9.

4 Experiments

In this section, we compare the uncertainty estimation methods proposed in the previous section, with the goal of answering the two research questions raised in Section 1. More specifically, we will compare RS that combine MF and MLP presented in Section 2 with the uncertainty estimators discussed in Section 3. In Table 1, we list all the models and uncertainty estimation methods that we consider.

4.1 Datasets

We evaluate our models and uncertainty estimation methods on two popular datasets: an implicit version of the Movielens 1M dataset [14][3] and one Pinterest dataset [8]. Table 2 presents some summary statistics.

For both datasets, we use a user-based random data splitting method: 60% of the interactions for each user are for training, 20% for validation and 20% for testing.

[2] 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.

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

Table 2. Datasets we use.

Dataset	Users	Items	Interactions
Movielens	6040	3416	1 Million
Pinterest	55187	9643	1.5 Million

4.2 Tuning

MF and MLP have hyperparameters that need to be chosen. First, we set the user and item latent embeddings size D to 128. 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 [28], $D = 128$ gives us reasonable computational cost. Furthermore, to suppress the need to tune the number of training iterations, we employ early-stopping to end the learning phase when the MAP@5 (see Eq. 13) on the validation set does not improve for three consecutive iterations.

We tuned our models using a Bayesian parameter search, assisted by Optuna [1]. We train each model 20 times by sampling the hyperparameters from the following:

- For all models, learning rate is sampled from $[0.0001, 0.01]$ and N, the number of negative training instances per positive training instance, from $\{1, 2, ..., 20\}$.
- MF: The L2 penalty factor applied to the user and item factors was taken from $[10^{-6}, 10^{-4}]$.
- MLP: We use the same three-layer MLP as in [15]. We also employ 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 hyperparameters related to it across the exact same grid as used in [3].
- Ensemble and sample sizes: We use $n = 5$ in Eqs. 8 – 9. We experimented with larger and smaller sample sizes, but found they all produced similar results.

4.3 Evaluation

To evaluate a model's recommendations, we obtain the top-K recommendation list for each user, which we denote by Z_u^K. These are the K candidate items that have highest predicted relevance score r_{ui} for the user. Candidate items exclude those that the user has interacted with in the training and validation sets; candidates are therefore items that either the user has not interacted with or items that the user has interacted with but the user's interaction with the item is recorded in the test set.

Let rel_u be the items that u has interacted with that are in the test set. Then, we evaluate a recommendation list according to its Mean Average Precision (MAP), averaged over all users:

$$\text{MAP@K} = \frac{1}{\#U} \sum_{u \in U} \sum_{j=1}^{K} \text{Precision@j}_u \times \delta\left(Z_u^K(j) \in \text{rel}_u\right) \tag{13}$$

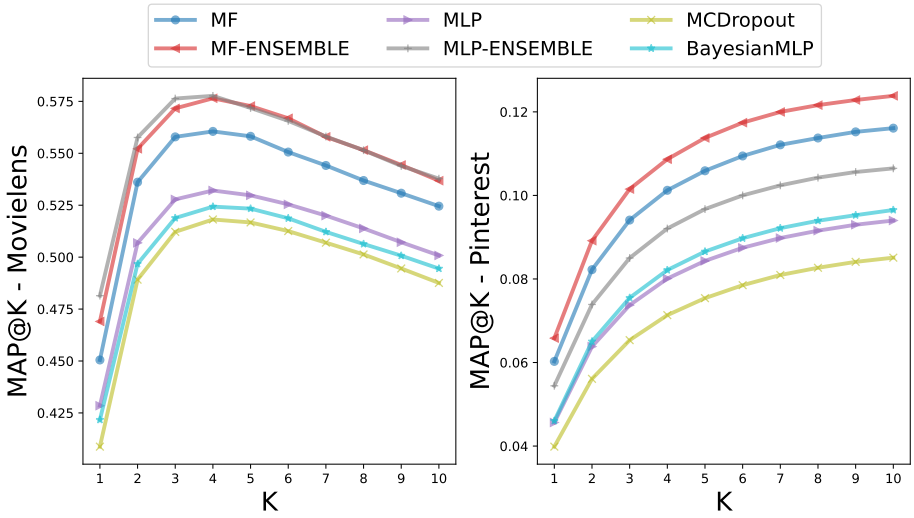

Fig. 1. $MAP@K$ for $K = 1, 2 \ldots, 10$ for the Movielens (left) and Pinterest (right) datasets.

where

$$\text{Precision@j}_u = \frac{\#\{Z_u^j \cap \text{rel}_u\}}{j} \tag{14}$$

and where $\delta(Z_u^K(j) \in \text{rel}_u) = 1$ if $Z_u^K(j)$, which is the j-th item in Z_u^K, is in rel_u and 0 otherwise.

4.4 Results

To answer **Q1** from Section 1, we compute the accuracy of the top-K recommendations for different recommendation list sizes $K = 1, 2 \ldots, 10$. Figure 1 shows the $MAP@K$ obtained in both datasets. Note that MF-NUS & MF-NIS and MLP-NUS & MLP-NIS are omitted because their MAP is the same as MF or MLP.

The ensemble models, MF-ENSEMBLE and MLP-ENSEMBLE, show a remarkable MAP improvement over the baselines, MF and MLP. On the other hand, the BayesianMLP has similar performance to the deterministic MLP, and MCDropout has the worst performance on both datasets. Therefore, we found that some models that perform uncertainty estimation improve accuracy, others worsen it. Clearly, ensembling emerges as the most beneficial method with respect to accuracy.

To answer **Q2**, we analyze the accuracy of the models for users, grouped according to their average recommendation uncertainty. More precisely, we calculated the average uncertainty on each user's recommendation list, and split the users into 10 equal-sized uncertainty bins, where bin 1 will have the 10% of users with the smallest average recommendation uncertainty and bin 10 will have those with the highest. Our intuition is that accuracy will fall as uncertainty grows. Figure 2 shows the results.

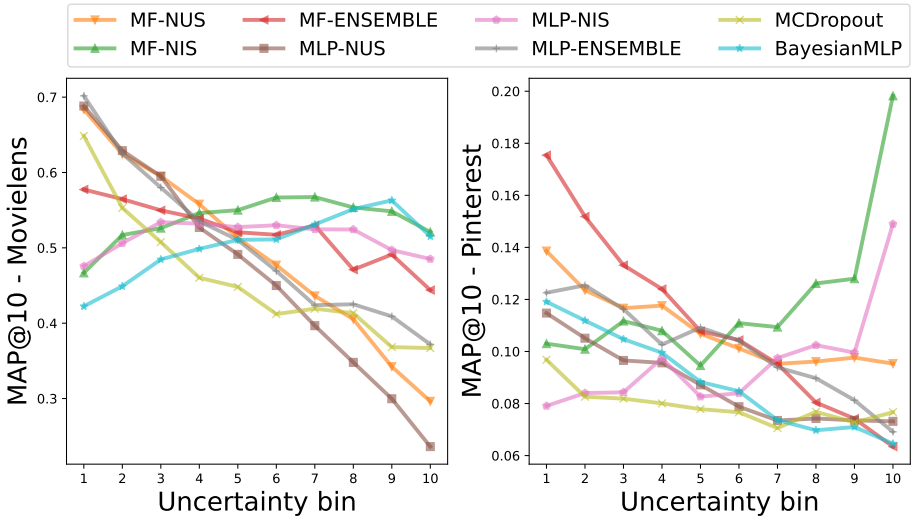

Fig. 2. *MAP@*10 for users grouped according to the average uncertainty of their recommendation. The higher the bin index, the higher is uncertainty.

In line with our intuition, we see that MAP has a strong negative correlation with some of the uncertainty estimation methods. In particular, MF-Ensemble, MLP-Ensemble, MF-NUS and MLP-NUS appear to be those more strongly reflecting the expected behaviour. The results for NEG-USER-SUPPORT show that mature users tend to get more accurate recommendations. In fact, NEG-USER-SUPPORT has the strongest correlation to MAP in the Movielens dataset. On the other hand, in the Pinterest data, the MF-Ensemble is the one to achieve the strongest correlation. This, together with the earlier results in Figure 1, show that ensembling is not only a technique that can boost the accuracy of recommenders, but can also offer uncertainty estimates that correspond with the expected recommendation accuracy.

Other models to follow the expected behaviour are MCDropout in both datasets, and the BayesianMLP in the Pinterest dataset. Oddly, the BayesianMLP shows a growing MAP curve in the Movielens data, meaning that users with higher recommendation list uncertainties are getting higher accuracy, which is a result that needs further investigation.

Models using NEG-ITEM-SUPPORT do not show a very strong correlation between uncertainty and MAP with the exception of the last uncertainty bins in the Pinterest case. This too is a result that needs further investigation.

Largely, we believe that we have obtained a positive answer to **Q2**. For this research question, ensembling has, again, proven to be a great tool. Nevertheless, the simple and cheap NEG-USER-SUPPORT metric can also provide good value with no computational cost added, in contexts where user-centric estimates suffice.

5 Related Work

In this section, we briefly describe some related work that could be further explored in RS research.

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 [2]. 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 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 [6], Gaussian embeddings are learned by a matrix factorization algorithm; in [18], 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, they do not quantify the uncertainty of item relevance to a user.

Neupane et al. [24] propose a method for quantifying the amount of evidence available when providing recommendations to cold-start users. They propose a meta-evidential method for doing so. We believe that, in future research, uncertainty could be inferred in a similar way.

Finally, in the related field of Information Retrieval, Penha & Hauff explore uncertainty in neural learning-to-rank models [26]. They obtain uncertainty estimates for a BERT ranker with the usage of Monte-Carlo Dropout and Deep-Ensembles, which we explained earlier. 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 [12]. However, their results are largely negative: they did not find this form of ranking to be more accurate. We leave the exploration of uncertainty-aware recommendation strategies similar to these for future research.

6 Conclusion

In this work, we explored methods for uncertainty estimation for implicit feedback recommender systems, exploring how the uncertainty estimates affect accuracy (**Q1**) and intelligibility regarding the recommender accuracy (**Q2**). Some of the methods had a positive impact on accuracy, others a negative impact. In particular, ensembling was the method showing the greatest accuracy improvements. Similarly,

ensembling also was one of the top contenders when it came to correlation between accuracy and uncertainty, together with NEG-USER-SUPPORT, suggesting that these methods can help to identify which users are prone to receive the most or least accurate recommendations.

In addition, in the previous section, we highlighted some related work that can be useful for further exploration in the area. We hope that these, together with the promising results shown by our experiments will foment new research in this largely unexplored field.

References

1. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A next-generation hyperparameter optimization framework. In: Procs. of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2019)
2. Bernardis, C., Ferrari Dacrema, M., Cremonesi, P.: Estimating Confidence of Individual User Predictions in Item-based Recommender Systems. In: Procs. of the 27th ACM Conference on User Modeling, Adaptation and Personalization. pp. 149–156 (2019)
3. Blundell, C., Cornebise, J., Kavukcuoglu, K., Wierstra, D.: Weight uncertainty in neural networks. In: Procs. of the 32nd International Conference on Machine Learning. pp. 1613–1622 (2015)
4. Cheng, H.T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., Ispir, M., et al.: Wide & deep learning for recommender systems. In: Procs. of the 1st Workshop on Deep Learning for Recommender Systems. pp. 7–10 (2016)
5. D'Amico, E., Gabbolini, G., Bernardis, C., Cremonesi, P.: Analyzing and improving stability of matrix factorization for recommender systems. Journal of Intelligent Information Systems **58**, 255—285 (2022)
6. Dos Santos, L., Piwowarski, B., Gallinari, P.: Gaussian embeddings for collaborative filtering. In: Procs. of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 1065–1068 (2017)
7. Gal, Y., Ghahramani, Z.: Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In: Procs. of the 33rd International Conference on Machine Learning. pp. 1050–1059 (2016)
8. Geng, X., Zhang, H., Bian, J., Chua, T.S.: Learning image and user features for recommendation in social networks. In: Procs. of the IEEE International Conference on Computer Vision. pp. 4274–4282 (2015)
9. Geyer, C.J.: Practical Markov Chain Monte Carlo. Statistical Science pp. 473–483 (1992)
10. Goan, E., Fookes, C.: Bayesian neural networks: An introduction and survey. In: Case Studies in Applied Bayesian Data Science, pp. 45–87. Springer (2020)
11. Graves, A.: Practical variational inference for neural networks. In: Procs. of the 24th International Conference on Neural Information Processing Systems. pp. 2348—2356 (2011)
12. Guiver, J., Snelson, E.: Learning to Rank with SoftRank and Gaussian Processes. In: Procs. of the 31st International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 259–266 (2008)
13. Guo, H., Tang, R., Ye, Y., Li, Z., He, X.: DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. In: Procs. of the International Joint Conference on Artificial Intelligence. pp. 1725–1731 (2017)
14. Harper, F.M., Konstan, J.A.: The Movielens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems **5**(4), 1–19 (2015)

15. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.S.: Neural collaborative filtering. In: Procs. of the 26th International Conference on the World Wide Web. pp. 173–182 (2017)
16. Hernando, A., Bobadilla, J., Ortega, F., Tejedor, J.: Incorporating reliability measurements into the predictions of a recommender system. Information Sciences **218**, 1–16 (2013)
17. Hu, Y., Koren, Y., Volinsky, C.: Collaborative filtering for implicit feedback datasets. In: Procs. of the 8h IEEE International Conference on Data Mining. pp. 263–272 (2008)
18. Jiang, J., Yang, D., Xiao, Y., Shen, C.: Convolutional gaussian embeddings for personalized recommendation with uncertainty. In: Procs. of the 28th International Joint Conference on Artificial Intelligence. pp. 2642–2648 (2020)
19. Kendall, A., Gal, Y.: What uncertainties do we need in Bayesian deep learning for computer vision? In: Procs. of the 31st International Conference on Neural Information Processing Systems. pp. 5580–5590 (2017)
20. Lakshminarayanan, B., Pritzel, A., Blundell, C.: Simple and scalable predictive uncertainty estimation using deep ensembles. In: Procs. of the 31st International Conference in Neural Information Processing Systems. pp. 6405–6416 (2017)
21. Liang, D., Krishnan, R.G., Hoffman, M.D., Jebara, T.: Variational autoencoders for collaborative filtering. In: Procs. of the 2018 World Wide Web Conference. pp. 689–698 (2018)
22. Mazurowski, M.A.: Estimating Confidence of Individual Rating Predictions in Collaborative Filtering Recommender Systems. Expert Systems with Applications **40**(10), 3847–3857 (2013)
23. Naumov, M., Mudigere, D., Shi, H.J.M., Huang, J., Sundaraman, N., Park, J., Wang, X., Gupta, U., Wu, C.J., Azzolini, A.G., et al.: Deep Learning Recommendation Model for Personalization and Recommendation Systems. arXiv:1906.00091 (2019)
24. Neupane, K.P., Zheng, E., Yu, Q.: MetaEDL: Meta Evidential Learning For Uncertainty-Aware Cold-Start Recommendations. In: Pros. of the IEEE International Conference on Data Mining. pp. 1258–1263 (2021)
25. Peña, F.J., O'Reilly-Morgan, D., Tragos, E.Z., Hurley, N., Duriakova, E., Smyth, B., Lawlor, A.: Combining Rating and Review Data by Initializing Latent Factor Models with Topic Models for Top-N Recommendation. In: Procs. of the 14th ACM Conference on Recommender Systems. pp. 438–443 (2020)
26. Penha, G., Hauff, C.: On the Calibration and Uncertainty of Neural Learning to Rank Models. arXiv:2101.04356 (2021)
27. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: Bpr: Bayesian personalized ranking from implicit feedback. arXiv preprint arXiv:1205.2618 (2012)
28. Rendle, S., Krichene, W., Zhang, L., Anderson, J.: Neural collaborative filtering vs. matrix factorization revisited. In: Procs. of the 14th ACM Conference on Recommender Systems. pp. 240–248 (2020)
29. Wang, C., Liu, Q., Wu, R., Chen, E., Liu, C., Huang, X., Huang, Z.: Confidence-Aware Matrix Factorization for Recommender Systems. In: Procs. of the 32nd AAAI Conference on Artificial Intelligence. pp. 434—-442 (2018)
30. Xiao, Y., Wang, W.Y.: Quantifying uncertainties in natural language processing tasks. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 7322–7329 (2019)
31. Zhu, B., Ortega, F., Bobadilla, J., Gutiérrez, A.: Assigning reliability values to recommendations using matrix factorization. Journal of Computational Science **26**, 165–177 (2018)