Revisiting Neighbourhood-Based Recommenders For Temporal Scenarios

Alejandro Bellogín
Universidad Autónoma de Madrid
Madrid, Spain
alejandro.bellogin@uam.es

Pablo Sánchez
Universidad Autónoma de Madrid
Madrid, Spain
pablo.sanchezp@estudiante.uam.es

ABSTRACT

Modelling the temporal context efficiently and effectively is essential to provide useful recommendations to users. Methods such as matrix factorisation and Markov chains have been combined recently to model the temporal preferences of users in a sequential basis. In this work, we focus on Neighbourhood-based Collaborative Filtering and propose a simple technique that incorporates interaction sequences when producing a personalised ranking. We show the efficiency of this method when compared against other sequence- and time-aware recommendation methods under two classical temporal evaluation methodologies.

1 INTRODUCTION

Recommmender Systems have become a necessary tool for a large number of online applications because of their ability to make personalised recommendations by adapting to user profiles. They are widely implemented in many online commercial platforms like Amazon, Netflix, Youtube, etc. Each of them can use different approaches like collaborative filtering, content-based, and hybrid approaches. Since the purpose of these systems is to provide the best possible suggestions, different types of information can be added to the recommendations (location, type of product, time, ...), also known as contextual information. Among the different contexts, temporal information is one of the most interesting contexts to be integrated into the recommendation algorithms, due to its facility to be captured and because it usually discriminates better than other dimensions [1]. Nonetheless, its formalisation in the area has typically been proposed as heuristic filters, both in terms of algorithmic approaches or evaluation strategies [4].

One of the earliest and most popular collaborative filtering approaches is the neighbourhood-based recommender, either user-based or item-based (in this article we will focus on the user-based variation). These approaches are normally represented as an aggregation function of the ratings from the k most similar users over item i; usually, this aggregation function is represented as [6]:

$$\hat{r}_{ui} = \frac{\sum_{v \in \mathcal{N}_i(u)} r_{vi} w_{uv}}{\sum_{v \in \mathcal{N}_i(u)} |w_{uv}|} \quad (1)$$

where w_{uv} is the weight (or similarity) between users u and v and $\mathcal{N}_i(u)$ represents user's u neighbours that have rated item i. Different normalisation functions can be applied to this formula, like mean centering or Z-score [11, 18].

In this work, we generalise this classical formulation – borrowing ideas from Aggregated Search and Information Retrieval in general [2] – to one where each neighbour provides a list of suggestions for each user, which are later combined into a single ranking. Under this perspective, modelling the temporal aspect of user preferences is straightforward – as we shall show here – and provides an intuitive rationale about what is being recommended and why.

In the remaining of this paper, we will answer the following research questions: (**RQ1**) Are neighbourhood-based recommenders competitive in temporal scenarios, especially when compared against methods based on matrix factorisation or Markov chains? (**RQ2**) Is there any advantage in using a rank aggregation formulation for this problem? Furthermore, how can we incorporate temporal sequences into this formulation?

After presenting our proposed method in detail in the next section, we address the research questions experimentally on real interactions from the Epinions website, using two evaluation methodologies to derive the temporal split. As we shall see, the empirical results validate our approach, showing performance improvements over state of the art memory-based alternatives and recent algorithms specifically tailored for sequential recommendation.

2 INTEGRATING TEMPORAL SEQUENCES IN NEIGHBOUR-BASED RECOMMENDERS

Neighbourhood-based recommenders can be revisited as ranking fusion algorithms where each neighbour contributes a ranking (of potential relevant items for the target user) and the goal of the recommender system is to combine these rankings into one final output. In terms of Aggregated Search [8, 15], each neighbour would be denoted as a *judge* (in Information Retrieval these judges are usually different search engines) who gives a complete ordering of all the alternative items to be ranked; each of these rankings is denoted as τ, and the final fused ranking is $\hat{\tau}$. Formally, the process of rank aggregation is divided into normalisation (where the scores or the ranks of τ are normalised into a common scale, $w^\tau(i)$, for each item i) and combination (where the normalised weights $w^\tau(i)$ are combined into one fused score).

There are several methods for each of these stages, see [15] for an in-depth review of the most prominent ones. Interestingly, by taking the identity normaliser for the scores ($w^\tau(i) = \tau(i)$) and the so-called CombSUM combiner (where the normalised weights are simply added for each item) with a preference weight for each ranking equals to the similarity between the neighbour and the target user, we obtain a linear combination of the normalised weights, which is equivalent to the classical formulation of a neighbourhood-based recommender. In fact, when we take into account the ratings of the neighbours, the "score" of user u to item i using CombSum and the identity normaliser produces the numerator of Equation 1. In this situation, each ranking τ is composed of the item-rating pairs rated by a particular neighbour, excluding, as a standard practice in

RecTemp '17, August 2017, Como, Italy

the community, those items already rated by the target user in training. Further extensions and ad-hoc modifications could be made to these normalisers and combiners so that other formulations of this problem – such as mean-centering or Z-score normalisation [6] – can be achieved.

Once we have reformulated the problem of neighbourhood-based recommendation as a ranking fusion technique, we now describe how we can incorporate temporal information in the process. The main idea is that **each neighbour will find which is her last common interaction with the target user and will create a ranking of her candidate alternatives iterating around that item**, taking into account the order in which she rated each of those alternatives. Although in this case we are not taking into account the actual moment of the interaction (i.e., we can end up recommending items from a neighbour whose last common item was rated long time ago), we can easily improve this approach by filtering neighbours whose last common item was rated before a certain threshold date; in this paper we will not explore this option and leave it as future work.

Note that the temporal aspect is considered twice in this model: it is used to involve the target user (through the last common interaction) in setting the actual moment (context) of the recommendation and, at the same time, to exploit the actual (temporal) order in which the neighbour interacted with the items. In the following, we present our model, consisting in a method to compute the last common interaction and different strategies to exploit the order of the neighbour ratings.

The last common interaction between two users u and v is:

$$n^*(u;v) = \max_k \left(i_k \in \mathcal{I}_u^t : i_k \in \mathcal{I}_v^t\right) \tag{2}$$

where $\mathcal{I}_u^t$ are the items rated by user u ordered by timestamp in ascending order (recent interactions appear later in the list), that is:

$$\mathcal{I}_u^t = \text{sort}\left(\mathcal{I}_u, t\right) = \left(i_k^t\right)_{k=1}^{|\mathcal{I}_u|}, \text{ with } t\left(i_k^t\right) < t\left(i_{k+1}^t\right) \tag{3}$$

Note that the last common interaction n^* will not be symmetrical in general – that is, $n^*(u;v) \neq n^*(v;u)$ – since it makes reference to the preferences of the first user.

Once we have sorted the preferences by timestamp ($\mathcal{I}_u^t$ and $\mathcal{I}_v^t$) and calculated the last common interactions ($n^*(u;v)$ and $n^*(v;u)$) for target user u and neighbour v, we propose three strategies to build the lists with candidate items from each neighbour: (a) taking the most recent m items rated by the neighbour after the last common interaction (we denote this list as $L_m^+(v)$ and the strategy as forward or F), (b) taking the most recent m items rated before the last common interaction ($L_m^-(v)$, backward or B), and (c) concatenating the m_1 items rated before and the m_2 items rated after the last common interaction ($L_{m_1,m_2}^{\pm}(v)$, backward-forward or BF). More specifically, these lists are generated as follows:

$$\text{Let } \mathcal{I}^t(v;u) = \text{sort}\left(\mathcal{I}_v - \mathcal{I}_u, t\right)$$

$$L_m^+(v) = \left(i_k^t\right)_{n^*}^{n^*+m}, i_k^t \in \mathcal{I}^t(v;u) \tag{4}$$

$$L_m^-(v) = \left(i_k^t\right)_{n^*-m}^{n^*}, i_k^t \in \mathcal{I}^t(v;u) \tag{5}$$

$$L_{m_1,m_2}^{\pm}(v) = \left(L_{m_1}^+(v), L_{m_2}^-(v)\right) \tag{6}$$

Therefore, for each neighbour we obtain a list $L(v)$ with all the candidate items from that neighbour, which will be later normalised and combined, as explained before, to produce a single ranking, containing the recommendations for the target user.

Figure 1: Example of user interactions in the movie domain. The yellow border corresponds to the last common interaction between u and each neighbour, the red border represents those movies included in $L_2^-(v)$, and the green border those in $L_2^+(v)$.

In summary, when using this formalisation, we obtain a model equivalent to classical formulations that can further incorporate the temporal information under different models.

Finally, let us illustrate the whole process with an example shown in Figure 1 using the movie domain. For the sake of simplicity, we do not include the user's rating, so the reader should assume that all sequences are composed of articles that the user has equally liked. In the case of movies, the temporal component is usually determinant, since newer movies tend to be consumed more often than older ones. In the presented example, user u is the user to whom we want to make the recommendations, and we represent three neighbours v_1, v_2, and v_3, where v_1 and v_3 have 3 items in common whereas v_2 shares 4 items with the target user. According to these interactions, the candidate items generated with respect to the different strategies presented before (limited to size 2) will be (considering that $n^*(v_1;u) = i_9, n^*(v_2;u) = i_{10}, n^*(v_3;u) = i_7$):

$$L_2^+(v_1) = (i_{14}, i_{13}), L_2^-(v_1) = (i_6, i_2), L_{1,1}^{\pm}(v_1) = (i_{14}, i_6)$$
$$L_2^+(v_2) = (i_{12}, i_{13}), L_2^-(v_2) = (i_2), L_{1,1}^{\pm}(v_2) = (i_{12}, i_2)$$
$$L_2^+(v_3) = (i_{12}, i_{15}), L_2^-(v_3) = (i_5, i_6), L_{1,1}^{\pm}(v_3) = (i_{12}, i_5)$$

Let us now consider that items i_{12}, i_{13} and i_{14} are in the test set (as mentioned before, newer films are more likely to be chosen by user u). A standard neighbourhood-based recommender which does not take the temporal aspect into account would probably recommend item i_2 whereas this item only appear in our approach once for strategy $L^{\pm}$ and twice for L^-, mostly in favour of the more recent movie i_{12}. Moreover, we believe that moving forward from the last common interaction is more useful in terms of recommendation performance – especially for novelty purposes –; this is evidenced by the strategies L^+ and $L^{\pm}$ that recommend i_{13} and i_{14}.

3 EMPIRICAL EVALUATION

3.1 Evaluation Methodology

We test the proposed approach on a dataset collected from *Epinions.com* by the authors of [21], also used in [10]. It includes all actions of all users on the website spanning January 2001 to November 2013, for a total of 193,571 actions on 42,447 items by

117,323 users; it hence has a density of 0.004%.[1] This dataset fits naturally the purpose of exploiting the temporal dimension of the user preferences, since it represents an unbiased sample of the website; other datasets more common in the area – like MovieLens – are not well-suited because they have been filtered or their timestamps are artificial [9].

As described in [4], there are several evaluation conditions worth of exploration when evaluating time-aware recommender systems. In this work, we use a time-dependent rating order (the timestamps of the test split for each user occur after those of the training split) in two evaluation methodologies: one with a user-centered base set and a fixed size condition (the last 2 actions of each user with at least 4 actions are included in the test split) and another with a community-centered base set and a proportion-based size condition (the same timestamp is used for all the users, in such a way that we retain the data corresponding to the 80% of the most recent rating times for training, and the rest for testing). We name the first configuration as *Fix* and the second as *CC*. There are obvious differences between these two evaluation methodologies: whereas in *CC* the test set is always (for every user) after the training set, in *Fix* this may not be the case; besides, (almost) every user will be included in the test set of *Fix* and this will not be the case in *CC*. In other terms, the *CC* methodology represents better a real environment, where there are some users that may not be active at some point, whereas with the *Fix* evaluation we can analyse the recommendations for all the users, not only the most active (or active in the last period of time) ones.

Using the terminology in [19], we report the results obtained following the *TrainItems* strategy to select the candidate items to be ranked by each algorithm; that is, a ranking is generated for each user by predicting a score for every item that has a rating in the training set. We then compute standard Information Retrieval metrics on the ranking, considering as relevant every item rated with a 5 in the test split. We report here the values for precision and recall at 5 and 50, and nDCG at 5 and 10. We also report the *user space coverage* metric (cvg) as defined in [20], that is, the number of users for which the system is able to recommend at least one item. Complete code and evaluation scripts can be found in the following Bitbucket repository: PabloSanchezP/BFRecommendation.

3.2 Recommendation algorithms

We compare our methods against different well-known state-of-the-art recommenders. We report a popularity-based recommender (ItemPop) that recommends items based on their popularity in the system. We also include a classical nearest-neighbour recommender optimised for ranking (that is, without normalisation, as proposed in [5]) using the Jaccard coefficient as similarity (KNN). A modification of this algorithm is also tested including an exponential time decay weight as introduced in [7] (TD). These three algorithms were based on the implementation found in the RankSys[2] library.

Additionally, we include some purely sequential-based algorithms as a comparison with the results reported in [10] (we use the same implementation as the one used in that paper). In the model named FMC (Factorised Markov Chains) the item-to-item transition matrix is factorised to capture the likelihood that an arbitrary user transitions from one item to another, using a first-order Markov chain. As an extension to this method, we include Factorised Personalized Markov Chains (FPMC), a method that combines Matrix Factorisation and first-order Markov Chains [17]. Finally, we also include as a baseline the Fossil method introduced in [10] (Factorised Sequential Prediction with Item Similarity Models), where Markov chains are combined with a similarity-based algorithm.

We compared these baselines against different instantiations of the rank aggregation formulation presented in Section 2. For the sake of space, we only report results when the backward-forward (BF) strategy to build the lists with candidate items is used, mostly due to its better performance with respect to the backward and forward strategies. We do experiment with a variation where the similarity is used to weight the contribution of each neighbour (as in standard user-based CF) and denote it as BFwCF; when no weight is used in the combination step we denote it as BFuCF.

Unless stated otherwise, we use 100 neighbours in KNN and TD, a λ factor of $1/200$ in TD, and $L = 1, K = 10, \lambda_\Theta = 0.1$, and $\alpha = 0.2$ in FMC, FPMC, and Fossil, as specified in the original paper [10] for this dataset; that is, we have not performed an exhaustive search to find the optimal parameters of these algorithms.

3.3 Results and Discussion

As described previously, we test our approaches under two evaluation conditions that consider differently the temporal dimension when splitting the dataset into training and test. As shown in Table 1, the *CC* methodology is slightly more difficult than *Fix*, as evidenced by the lower values obtained in most of the metrics by the baseline algorithms. This observation is in line with previous results in the area [4]. It should be noted that *CC* replicates a situation closer to what we would find in an online experiment, where the test split is set in the future no matter the user we are considering; this is not true in *Fix*, where the test split of each user could exist at different timestamps, but on the other hand, every user contains the same number of test interactions, which may decrease the bias towards users more active in the most recent portion of the dataset.

We now assess the research questions RQ1 (Are neighbourhood-based recommenders competitive in temporal scenarios, especially when compared against methods based on matrix factorisation or Markov chains?) and RQ2 (Is there any advantage in using a rank aggregation formulation for this problem?) raised at the beginning of the paper, in light of the results summarised in Table 1. To address **RQ1** we compare the performance of FMC, FPMC, and Fossil (combinations of Markov chains and matrix factorisation) against the KNN baseline. We observe that, in contrast to the original paper [10], Fossil is not always the best performing method amongst these baselines (this only holds when using the *CC* methodology). Actually, the results obtained for these methods are worse than KNN in both methodologies. We argue that a possible reason for this inconsistency with respect to the previous reported results is that here we evaluate using a more common setting in the area (top-N recommendation) and not AUC, which these algorithms are optimised for. Furthermore, in [10] no classical recommender algorithms like KNN appear in their comparison.

Furthermore, it is interesting to note that a time decay modification of KNN does not improve under this setting unless many items are considered in the ranking, since TD outperforms KNN only for Precision@50 and Recall@50. In any case, it seems that basic KNN algorithms are competitive against state-of-the-art algorithms specifically designed to address the sequential recommendation

[1] Note these statistics do not match those from [10] or [21] because we do not put any constraint on the minimum number of ratings on users and items.
[2] http://ranksys.org

Table 1: Summary of comparative effectiveness, including improvement (Δ) in terms of nDCG@5 with respect to two of the baselines. Best values for each evaluation methodology and metric are in bold.

(a) *CC* methodology

Method	Precision@5	nDCG@5	Recall@5	nDCG@10	Precision@50	Recall@50	cvg	Δ wrt KNN	Δ wrt Fossil
ItemPop	1.81E-04	8.89E-04	2.25E-03	1.21E-03	**3.80E-04**	**4.69E-02**	100.00%	-144.08%	-36.88%
KNN	2.29E-04	2.17E-03	2.17E-03	2.94E-03	4.59E-05	4.34E-03	100.00%	–	43.92%
TD	2.29E-04	2.17E-03	2.17E-03	2.17E-03	6.88E-05	6.51E-03	100.00%	0.00%	43.92%
FMC	0.00E+00	0.00E+00	0.00E+00	4.49E-04	2.69E-05	1.22E-03	85.21%	NA	NA
FPMC	0.00E+00	0.00E+00	0.00E+00	0.00E+00	2.69E-05	1.22E-03	85.21%	NA	NA
Fossil	2.69E-04	1.22E-03	2.43E-03	1.22E-03	2.69E-05	2.43E-03	85.21%	-78.31%	–
BFuCF	2.29E-04	2.17E-03	2.17E-03	2.17E-03	6.88E-05	4.49E-03	100.00%	0.00%	43.92%
BFwCF	**4.59E-04**	**3.10E-03**	**4.34E-03**	**3.10E-03**	9.17E-05	6.66E-03	100.00%	30.10%	60.80%

(b) *Fix* methodology

Method	Precision@5	nDCG@5	Recall@5	nDCG@10	Precision@50	Recall@50	cvg	Δ wrt KNN	Δ wrt Fossil
ItemPop	3.32E-04	1.28E-03	1.74E-03	2.12E-03	**4.06E-04**	**2.19E-02**	100.00%	-200.02%	-5.48%
KNN	**1.05E-03**	3.84E-03	**5.56E-03**	**4.94E-03**	3.83E-04	2.05E-02	97.20%	–	64.84%
TD	3.15E-04	1.09E-03	1.99E-03	1.62E-03	2.97E-04	1.68E-02	97.20%	-252.10%	-23.79%
FMC	4.34E-04	1.39E-03	2.42E-03	2.27E-03	2.91E-04	1.57E-02	100.00%	-176.32%	2.85%
FPMC	4.08E-04	1.08E-03	1.93E-03	1.67E-03	2.20E-04	1.10E-02	100.00%	-255.14%	-24.86%
Fossil	3.32E-04	1.35E-03	1.64E-03	2.28E-03	2.60E-04	1.38E-02	100.00%	-184.43%	–
BFuCF	**1.05E-03**	**3.89E-03**	**5.56E-03**	4.75E-03	3.62E-04	1.96E-02	97.20%	1.39%	65.33%
BFwCF	9.46E-04	3.48E-03	4.96E-03	4.65E-03	3.55E-04	1.91E-02	97.20%	-10.50%	61.15%

problem, even beating the ItemPop recommender, which is frequently a very strong baseline due to the inherent popularity bias found in this type of systems [12].

We now focus on the models generated using the rank aggregation formulation to address **RQ2**. We observe that these models (BFuCF and BFwCF) show very positive results in both evaluation methodologies. In fact, we experimented with several instantiations of the framework described in Section 2. We found that the best normalisation method is the identity normaliser, since a rank-based approach or the standard normaliser [15] produces worse results (not reported here because of space constraints). Our preliminary results also showed that the best strategy to select the candidate items is generating lists as $L^{\pm}_{m_1,m_2}$; because of this we report in Table 1 results for this strategy using $m_1 = m_2 = 5$, Jaccard as similarity, and 100 neighbours, so the comparison is as fair as possible with respect to the rest of the baselines, even though an exhaustive tuning of these parameters may achieve better performance values.

When the proposed approaches are used, a high improvement is achieved with respect to both KNN and Fossil baselines; however, depending on the actual evaluation methodology our approach may actually degrade its performance (see *Fix* methodology). In these cases, the coverage is limited by the coverage of the user similarity, which results in the same coverage as that obtained for KNN and TD algorithms. It is interesting to observe that the largest improvement is obtained for the more realistic scenario, that is, the *CC* methodology. Regarding the difference between similarity-weighted (BFwCF) and unweighted (BFuCF) versions of these methods, the conclusions are not clear, since this parameter seems to depend on how the split was performed. We hypothesise that the similarity values in the *CC* methodology are more meaningful because the timeline is the same for every user, even though the similarity used (Jaccard) does not take the temporal dimension into account.

4 CONCLUSIONS AND FUTURE WORK

In this paper we have presented a new formulation for neighbourhood-based recommendation that allows to integrate the temporal dimension seamlessly and successfully, according to the reported experiments. The two research questions proposed have been answered positively, evidencing that this type of recommendation algorithms is competitive in temporal scenarios, outperforming recent state-of-the-art algorithms specifically tailored to sequential-based recommendation. Furthermore, when formulating the recommendation problem as an aggregation of several rankings and introducing the temporal dimension in the process, the performance clearly improves, up to a 30% with respect to another neighbour-based recommender without using the temporal component and up to a 65% with respect to a sequential-based baseline.

Since the framework introduced in this work is general enough to work with other aggregation functions, in the future we plan to explore the behaviour of our proposal when alternative aggregation functions – such as those based on the score distribution [14] – are used. Furthermore, an exhaustive analysis – with more datasets, baselines such as SVD++ [13] and BPR for implicit data [16], and evaluation methodologies – should be made to better understand each component of the proposed model, for instance, the number of items allowed to be selected after and before the last common interaction and the impact of the similarity when weighting the final result. An important aspect that deserves further research is the definition of sequence-aware similarity metrics [3], so that the temporal dimension can be considered when selecting the neighbours in the proposed approach.

Acknowledgments

This work was funded by the national Spanish Government under project TIN2016-80630-P.

REFERENCES

[1] Gediminas Adomavicius and Alexander Tuzhilin. 2015. Context-Aware Recommender Systems. In *Recommender Systems Handbook*, Francesco Ricci, Lior Rokach, and Bracha Shapira (Eds.). Springer, 191–226. DOI: https://doi.org/10.1007/978-1-4899-7637-6_6

[2] Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. 2011. *Modern Information Retrieval - the concepts and technology behind search, Second edition.* Pearson Education Ltd., Harlow, England.

[3] Alejandro Bellogín and Pablo Sánchez. 2017. Collaborative Filtering based on Subsequence Matching: A New Approach. *Submitted to Information Sciences* (2017).

[4] Pedro G. Campos, Fernando Díez, and Iván Cantador. 2014. Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. *User Model. User-Adapt. Interact.* 24, 1-2 (2014), 67–119.

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

[6] Christian Desrosiers and George Karypis. 2011. A Comprehensive Survey of Neighborhood-based Recommendation Methods. In *Recommender Systems Handbook*. 107–144.

[7] Yi Ding and Xue Li. 2005. Time weight collaborative filtering. In *CIKM*. ACM, 485–492.

[8] Cynthia Dwork, Ravi Kumar, Moni Naor, and D. Sivakumar. 2001. Rank aggregation methods for the Web. In *Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001*, Vincent Y. Shen, Nobuo Saito, Michael R. Lyu, and Mary Ellen Zurko (Eds.). ACM, 613–622. DOI: https://doi.org/10.1145/371920.372165

[9] F. Maxwell Harper and Joseph A. Konstan. 2016. The MovieLens Datasets: History and Context. *TiiS* 5, 4 (2016), 19:1–19:19. DOI: https://doi.org/10.1145/2827872

[10] Ruining He and Julian McAuley. 2016. Fusing Similarity Models with Markov Chains for Sparse Sequential Recommendation. In *ICDM*. IEEE, 191–200.

[11] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. 1999. An Algorithmic Framework for Performing Collaborative Filtering. In *SIGIR '99: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 15-19, 1999, Berkeley, CA, USA*, Fredric C. Gey, Marti A. Hearst, and Richard M. Tong (Eds.). ACM, 230–237. DOI: https://doi.org/10.1145/312624.312682

[12] Dietmar Jannach, Lukas Lerche, Iman Kamehkhosh, and Michael Jugovac. 2015. What recommenders recommend: an analysis of recommendation biases and possible countermeasures. *User Model. User-Adapt. Interact.* 25, 5 (2015), 427–491. DOI: https://doi.org/10.1007/s11257-015-9165-3

[13] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD '08*. ACM, 426–434. DOI: https://doi.org/10.1145/1401890.1401944

[14] R. Manmatha, Toni M. Rath, and Fangfang Feng. 2001. Modeling Score Distributions for Combining the Outputs of Search Engines. In *SIGIR 2001: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, September 9-13, 2001, New Orleans, Louisiana, USA*, W. Bruce Croft, David J. Harper, Donald H. Kraft, and Justin Zobel (Eds.). ACM, 267–275. DOI: https://doi.org/10.1145/383952.384005

[15] M. Elena Renda and Umberto Straccia. 2003. Web Metasearch: Rank vs. Score Based Rank Aggregation Methods. In *Proceedings of the 2003 ACM Symposium on Applied Computing (SAC), March 9-12, 2003, Melbourne, FL, USA*, Gary B. Lamont, Hisham Haddad, George A. Papadopoulos, and Brajendra Panda (Eds.). ACM, 841–846. DOI: https://doi.org/10.1145/952532.952698

[16] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, June 18-21, 2009*, Jeff A. Bilmes and Andrew Y. Ng (Eds.). AUAI Press, 452–461. https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=1630&proceeding_id=25

[17] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized Markov chains for next-basket recommendation. In *WWW*. ACM, 811–820.

[18] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *CSCW '94, Proceedings of the Conference on Computer Supported Cooperative Work, Chapel Hill, NC, USA, October 22-26, 1994*, John B. Smith, F. Donelson Smith, and Thomas W. Malone (Eds.). ACM, 175–186. DOI: https://doi.org/10.1145/192844.192905

[19] Alan Said and Alejandro Bellogín. 2014. Comparative recommender system evaluation: benchmarking recommendation frameworks. In *RecSys*. ACM, 129–136.

[20] Guy Shani and Asela Gunawardana. 2011. Evaluating Recommendation Systems. In *Recommender Systems Handbook*. 257–297.

[21] Tong Zhao, Julian J. McAuley, and Irwin King. 2014. Leveraging Social Connections to Improve Personalized Ranking for Collaborative Filtering. In *CIKM*. ACM, 261–270.