Incorporating System-Level Objectives into Recommender Systems

Himan Abdollahpouri
Department of Information Science
University of Colorado Boulder
himan.abdollahpouri@colorado.edu

ABSTRACT

One of the most essential parts of any recommender system is personalization– how acceptable the recommendations are from the user's perspective. However, in many real-world applications, there are other stakeholders whose needs and interests should be taken into account. In this work, we define the problem of multi-stakeholder recommendation and we focus on finding algorithms for a special case where the recommender system itself is also a stakeholder. In addition, we will explore the idea of incremental incorporation of system-level objectives into recommender systems over time to tackle the existing problems in the optimization techniques which only look for optimizing the individual users' lists.

CCS CONCEPTS

• **Computing methodologies** → *Machine learning*.

KEYWORDS

Recommender Systems; Multistakeholder recommendation; Multi-objective recommendation

1 PROBLEM

Recommender systems provide personalized information access, supporting e-commerce, social media, news, and other applications where the volume of content would otherwise be overwhelming. They have become indispensable features of the Internet age, found in systems of many kinds. Even search engines, the most fundamental web applications, have become increasingly personalized in their provision of results, to the extent that they can also be considered recommender systems.

One of the defining characteristics of recommender systems is personalization. Recommender systems are typically evaluated on their ability to provide items that satisfy the needs and interests of the end user. Such focus is entirely appropriate–Users would not

make use of recommender systems if they believed such systems were not providing items that matched their interests. Still, it is also clear that, in many recommendation domains, the user for whom recommendations are generated is not the only stakeholder in the recommendation outcome. Other users, the providers of products, and even the system's own objectives may need to be considered. Fairness and balance are important examples of system-level objectives, and these social-welfare oriented goals may at times run counter to individual preferences. Sole focus on the end user hampers developers' ability to incorporate such objectives into recommendation algorithms and system designs.

The integration of the perspectives of multiple parties into recommendation generation and evaluation is the underlying goal of the new sub-field of *multistakeholder recommendation* [3, 6, 7, 11]. The goal of a recommender system in a multistakeholder environment is, therefore, to generate recommendations taking all the stakeholder's needs and preferences into account.

1.1 Multistakeholder Recommendation

In many recommender systems, there could exist a variety of different stakeholders. However, we can usually see three main stakeholders:

Consumers C: The consumers are those who receive the recommendations. They are the individuals whose choice or search problems bring them to the platform, and who expect recommendations to satisfy those needs.

Providers P: The providers are those entities that supply or otherwise stand behind the recommended objects and possibly gain utility from the consumer's choice.

System S: The final category is the platform itself, which has created the recommender system in order to match consumers with providers and has some means of gaining benefit from successfully doing so. The platform may be a retailer, e-commerce site, broker, or other venue where consumers seek recommendations.

Typically, the system will attempt to satisfy the needs of at least one type of stakeholder by offering recommendations tailored (at least in part) to their preferences. When the stakeholder is the consumer, this is the most common personalized recommendation scenario. When there is personalization for both the consumer and the provider, we have a reciprocal recommendation.

In many real-world contexts, the system may gain some utility when recommending items, which could be in the form of a simple aggregate of the other stakeholders' utilities. In many e-commerce settings, the system will get a commission for each sale, and such benefits can be considered together with personalization [11].

Alternatively, the system may seek to tailor outcomes specifically to achieve particular objectives. For example, an educational site may view the recommendation of learning activities as a curricular decision and seek to have its recommendations fit a model of student growth and development. Its utility may, therefore, be more complex than a simple aggregation of those of the other stakeholders. One way to think about a multistakeholder recommendation is to look at it as a two-stakeholder system: the user and the system. In other words, in many applications, it is possible to aggregate the utilities of other stakeholders within the system utility and therefore optimizing for the system utility would, indirectly, optimize for their utility as well. Therefore, in this work, we focus on a scenario where we have two main stakeholders, the end-user and the system.

In contrast to a single-stakeholder perspective which has been the case, for the most part, in recommender systems research, in this work we are interested in finding algorithms and mechanisms to incorporate system-level objectives into the recommendations.

That being said, there are several challenges that need to be investigated:

- How can the system-level preferences be incorporated into the recommendation generation?
- What are the different types of system-level preferences and how does the solution for incorporating each of them differs from the others?
- What is the right balance or trade-off between user preferences and system preferences? Does this balance vary depending on the domain?
- How can the system learn from the optimization in previous steps to do a better job at a particular time?

2 STATE OF THE ART

There is a large body of recent work on incorporating diversity, novelty, long tail promotion and other metrics as additional objectives for recommendation generation and evaluation. See, for example, [2, 15]. There is also a growing body of work on combining multiple objectives using constraint optimization techniques, including linear programming. See, for example, [8, 9]. These techniques provide a way to limit the expected loss on one metric (typically accuracy) while optimizing for another, such as diversity. The complexity of these approaches increases exponentially as more constraints are considered, making them a poor fit for the general multistakeholder case. Also, for the most part, multiobjective recommendation research concentrates on maximizing multiple objectives for a single stakeholder, the end user.

3 PROPOSED APPROACH

In this work, we will focus on one specific case of multistakeholder recommendation which is considering users and the system as two different stakeholders. That being said, we realized there is a substantial amount of work on using different types of non-accuracy metrics in recommender systems each of which used a different method to do the recommendation generation. For example, most of the work on recommendation list diversification, price optimization, fairness aware recommendation and so on use different algorithms to achieve the desired set of recommendation based on

what evaluation metrics they use. In other words, there is a lack of understanding of what all these different problems have in common or from what aspects they differ. What is missing is the need for a general way of looking at extra objectives (i.e. system objectives) in recommender systems and to define all different types of such objectives and develop algorithms which can be used in problems with a similar type of objectives. What we propose is to have a standard framework that could be the home for all these different problems. To start, we need to categorize different types of system-level objectives. One main categorization in terms of the type of optimization that could be used to solve such problems is based on the scope of the system objective. Therefore, we divide every system-level objective into two main groups:

Local objectives: in this group we have objectives that could be optimized within each user's recommendation list. In other words, there is no need to look at the entire user population in order to do the optimization. List diversification and novelty (from the user's perspective) belongs to this group.

Global objectives: The second group is the type of objectives that need to be taken care of not only based on the recommendations within each user's list but also the recommendation lists of other users as well. Objectives like long-tail promotion and fair exposure belong to this category.

In addition to this two classes of objectives, we can also define several other types of objectives based on whether the system's focus is on 1) items, 2) users or maybe 3) both at the same time. We define different types of system objectives with respect to which entity the system is focusing on as follow:

User-item related: The system could have a certain goal regarding which user should get what recommendation. This is more like a parental control which means, in addition to what users like, the system also has to decide whether that item suits the user or not. An example would be in educational recommendation where the system might not want to recommend a program to a user even though she likes it or, recommend a certain program even though she may not like it the most compared to other programs.

user related: When the system wants the users who receive a certain item (items) as the recommendation to have certain properties. For example, in an educational recommender system, the goal might be to make sure the users who receive the recommendation are %50 male and %50 female. Many of the fairness-aware recommendation problems fall in this category.

item related: This is when the system has a certain goal regarding a certain group of items. For example, long-tail recommendation, which is a well-known problem, is when the system wants to promote long-tail items more often.

4 RESULTS AND THE PROGRESS SO FAR

As a very specific case of system-level objective, we have worked on the problem of long-tail promotion (i.e. controlling popularity bias) in recommender systems. Popularity bias refers to the situation in which there is an over-concentration of the recommendations of certain items, the popular ones. Controlling popularity bias is important for many recommender systems as it affects the fairness

of the platform and it also improves the novelty of the recommendations. For more information on popularity bias see [1, 2]main. We wanted to go a bit deep in this problem and find different types of solutions for it to make a better sense of this type of objective and to be able to generalize for similar problems. Our aim is, however, to find a general class of solutions for other types of system objectives for the next steps of this research.

In order to control the effect of popularity bias in recommender systems we have used two approaches: One way is to modify the underlying recommendation algorithm such that we take the popularity of items into account in addition to the relevance of those items for generating the predicted ratings. Therefore, the final list is both personalized and also controlled for popularity bias.

Another approach is to keep the existing algorithm untouched but add an extra level of item filtering on the predicted ratings. In this method, the algorithm tries to re-rank the items based on both their accuracy (predicted rating) and popularity.

4.1 Technique 1: Model based Approach: Fairness-aware Regularization

In this approach, we have explored the use of regularization to control the popularity bias of a recommender system. We start with an optimization objective of the form:

$$\min_{P,Q} acc(P,Q) + \lambda reg(P,Q) \tag{1}$$

where $acc(.)$ is the accuracy objective, $reg(.)$ is a regularization term, and λ is a coefficient for controlling the effect of regularizer.

Our goal, therefore, is to identify a regularization component of the objective function that will be minimized when the distribution of recommendations is fair in terms of popular and non-popular items.

We define two sets of items (Γ') and (Γ) corresponding to the popular and non-popular items, and define a co-membership matrix D, over these sets. For any pair of items i and j, $d(i, j) = 1$ if i and j are in the same set and 0 otherwise. We define intra-list binary unfairness (ILBU) as the average value of $d(i, j)$ across all pairs of items i, j.

$$ILBU(L_u) = \frac{1}{N(N-1)} \sum_{i,j \in L_u} d(i,j) \tag{2}$$

where N is the number of items in the recommendation list. The fairest list is one that contains equal numbers of items from each set, which can be easily seen.

For more details on how the optimization is solved please refer to [2].

4.2 Technique 2: Post processing Re-ranking

4.2.1 xQuAD. For the second approach, we used a post-processing re-ranking technique to diversify the recommendation lists in terms of popular and unpopular items. We have used EXplicit Query Aspect Diversification (xQuAD) [12] which explicitly accounts for the various aspects associated with an under-specified query. In the context of search engines, items are selected iteratively by estimating how well a given document satisfies an uncovered aspect.

We build on the *xQuAD* model to control popularity bias in recommendation algorithms. We assume that for a given user u, a ranked recommendation list R has already been generated by a base recommendation algorithm. The task of the modified xQuAD method is to produce a new re-ranked list S ($|S| < |R|$) that manages popularity bias while still being accurate.

The new list is built iteratively according to the following criterion:

$$P(v|u) + \lambda P(v, S'|u) \tag{3}$$

where $P(v|u)$ is the likelihood of user $u \in U$ being interested in item $v \in V$, independent of the items on the list so far as predicted by the base recommender. The second term $P(v, S'|u)$ denotes the likelihood of user u being interested in an item v as an item not in the currently generated list S.

Intuitively, the first term incorporates accuracy while the second term promotes diversity between two different categories of items (i.e. short head and long tail). The parameter λ controls how strongly controlling popularity bias is weighted in general. The item that scores most highly under the equation 3 is added to the output list S and the process is repeated until S has achieved the desired length.

For more details on these two techniques refer to [2] and [4].

we tested our proposed algorithms on two public datasets. We show the result for one dataset due to lack of space but a, more or less, a similar result could be seen for the other dataset. The dataset for which you can see the result in figure 1, is the Epinions dataset which is gathered from a consumers opinion site where users can review items [10]. This dataset has the total number of 664,824 ratings given by 40,163 users to 139,736 items. We split the items in both datasets into two categories: long-tail (Γ) and short head (Γ')in a way that short head items correspond to %80 of the ratings while long-tail items have the rest of the %20 of the ratings.

4.3 Evaluation

The experiments compare four algorithms. Since we are concerned with ranking performance, we chose as our baseline algorithm RankALS, a pair-wise learning-to-rank algorithm. We used the output from RankALS as input for the re-ranking technique described above. Note that, we developed two versions of the re-ranking algorithm: Binary xQuAD and Smooth xQuAD, marked *Binary* and *Smooth* in the figures. The difference between the two as you can read in more details in [4] is in how they check for whether an item category (popular or non-popular) is already covered by the list or not. The binary version only returns 0 (not covered) or 1 (covered) while the smooth version returns what percentage of the list is covered with each item category. Also, the model-based approach is shown by *LT_Reg* in the figure. We compute lists of length 100 from RankALS and pass these to the re-ranking algorithms to compute the final list of 10 recommendations for each user. In order to evaluate the effectiveness of algorithms in mitigating popularity bias, we used four different metrics each of which measures a certain aspect of long-tail promotion.

Average Recommendation Popularity (ARP): This measure from [14] calculates the average popularity of the recommended items in each list. For any given recommended item in the list, we measure the average number of ratings for those items. More
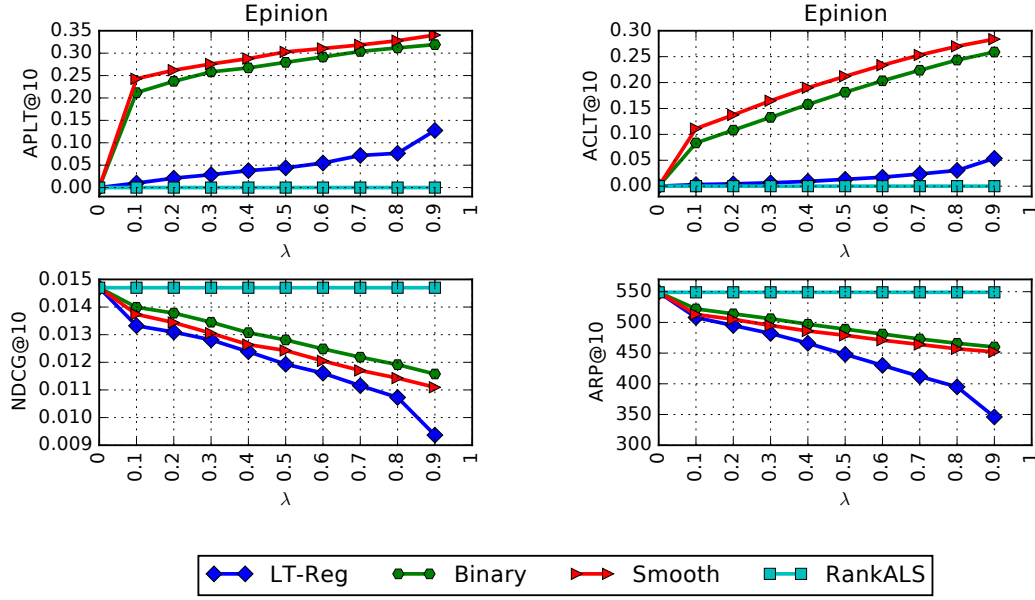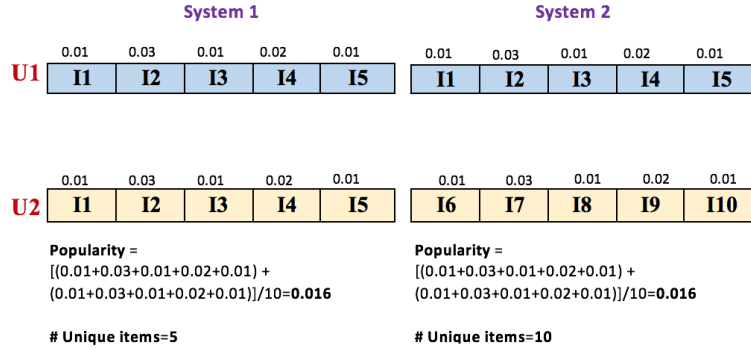
Figure 1: Results for the Epinions dataset

Figure 2: The recommendation lists of length 5 to different users U1 and U2 generated by two different recommender systems. Both systems have similar sum of popularity, but system 2 has covered 10 items while system 1 has only covered 5.

formally:

$$ARP = \frac{1}{|U_t|} \sum_{u \in U_t} \frac{\sum_{i \in L_u} \phi(i)}{|L_u|} \quad (4)$$

where $\phi(i)$ is the number of times item i has been rated in the training set. L_u is the recommended list of items for user u and $|U_t|$ is the number of users in the test set.

Average Percentage of Long Tail Items (APLT): As used in [2], this metric measures the average percentage of long tail items in the recommended lists and it is defined as follows:

$$APLT = \frac{1}{|U_t|} \sum_{u \in U_t} \frac{|\{i, i \in (L_u \cap \Gamma)\}|}{|L_u|} \quad (5)$$

This measure gives us the average percentage of items in users' recommendation lists that belong to the long tail set.

Average Coverage of Long Tail items (ACLT): This is the metric we introduced in [4] which measures how much exposure long-tail items get in the entire recommendations. One problem with *APLT* is that it could be high even if all users get the same set of long tail items. *ACLT* measures what fraction of the long-tail items the recommender has covered:

$$ACLT = \frac{1}{|U_t|} \sum_{u \in U_t} \sum_{i \in L_u} \mathbb{1}(i \in \Gamma) \quad (6)$$

where $\mathbb{1}(i \in \Gamma)$ is an indicator function and it equals to 1 when i is in Γ. This function is related to the *Aggregate Diversity* metric of [5] but it looks only at the long-tail part of the item catalog.

In addition to the aforementioned long tail diversity metrics, we also evaluate the accuracy of the ranking algorithms in order to examine the diversity-accuracy trade-offs. For this purpose we

5

use the standard *Normalized Discounted cumulative Gain* (*NDCG*) measure of ranking accuracy.

Figure 2 shows the results for the Epinions dataset across the different algorithms using a range of values for λ. (Note that the LT-Reg algorithm uses the parameter λ to control the weight placed on the long-tail regularization term.) All results are averages from five-fold cross-validation using a %80 -%20 split for train and test, respectively. As expected, the diversity scores improve for all algorithms, with some loss of ranking accuracy. Differences between the algorithms are evident, however. The exposure metric (ACLT) plot shows that the two re-ranking algorithms, and especially the Smooth version, are doing a much better job of exposing items across the long-tail inventory than the regularization method. The ranking accuracy shows that, as expected, the Binary version does slightly better as it performs minimal adjustment to the ranked lists. LT-Reg is not as effective at promoting long-tail items, either by the list-wise APLT measure or by the catalog-wise ACLT.

5 CONCLUSIONS AND FUTURE WORK

Working on tackling popularity bias as an example of system-level objective, we learned that the system objective could be incorporated in three different ways which we only explored two of them: 1) Model-based multi-objective optimization, 2) re-ranking approach, and 3) data manipulation which we have not yet worked on. The work we have done so far, might not be, just by itself, a great contribution to the field as there are similar approaches for tackling popularity bias but it inspired us to look at the problem in a more general way as popularity bias is not the only objective that a system could have. For example, *user-level list diversity*, *aggregate diversity*, *profit maximization*, *fairness-aware recommendation*, to name just a few, all could be considered as system-level objectives which should be incorporated into the recommendation process. So one future work is to find a general framework by which we can incorporate any type of system-level objectives into the recommender systems.

Another important observation we made which is also a possibility for future work is that almost all the works on long-tail promotion have done the optimization on individual's recommendation lists hoping this will lead to an overall optimization within the entire user base. However, as we saw in our experiments (ACLT vs ARP), a system could have a low overall sum of popularity for the recommendations but might not have done a good job of promoting enough long-tail items. The reason is optimizing the individual's recommendation list (local optimization) does not guaranty an overall optimization (global optimization) and, therefore, a more dynamic and temporal optimization technique is needed. Figure 2 shows a simple scenario where two systems have generated recommendations for two users. The popularity values for each item is shown on the top of each recommended item. A popularity value of 0.01 means the item has been rated by %1 of the users. As you can see, both systems have the same sum of popularity but one has covered 10 items while the other has only covered 5. This example emphasizes the need for better incorporation of system-level objectives into recommendations. One possibility to address this problem is to use incremental optimization [13] over time to balance system and user-level objectives. Figure 3 shows a schematic

view of how the optimization could be done over time such that, at any given time T, the recommender system optimizes the objective function by also looking at what the outcome for optimization was at previous times.
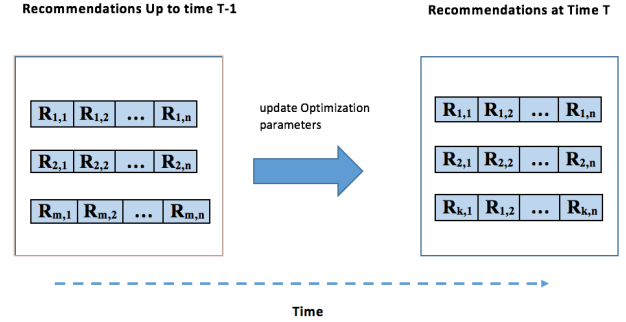

Figure 3: Dynamic optimization of the recommendations over time

REFERENCES

[1] Himan Abdollahpouri. 2019. Popularity Bias in Ranking and Recommendation. In *In AAAI/ACM Conference on AI, Ethics, and Society (AIES'19) January 27–28, 2019, Honolulu, HI, USA*. ACM.

[2] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. 2017. Controlling Popularity Bias in Learning to Rank Recommendation. In *Proceedings of the 11th ACM conference on Recommender systems*.

[3] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. 2017. Recommender Systems as Multistakeholder Environments. In *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization (UMAP'17)*. Bratislava, Slovakia, To appear.

[4] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. 2019. Managing Popularity Bias in Recommender Systems with Personalized Re-ranking.. In *FLAIRS Conference*. To appear.

[5] G. Adomavicius and Y.O. Kwon. 2012. Improving aggregate recommendation diversity using ranking-based techniques. *Knowledge and Data Engineering, IEEE Transactions on* 24, 5 (2012), 896–911.

[6] Robin Burke and Himan Abdollahpouri. 2016. Educational Recommendation with Multiple Stakeholders. In *Third International Workshop on Educational Recommender Systems*. Omaha.

[7] Robin Burke, Himan Abdollahpouri, Bamshad Mobasher, and Trinadh Gupta. 2016. Towards Multi-Stakeholder Utility Evaluation of Recommender Systems. In *Proceedings of the International Workshop on Surprise, Opposition, and Obstruction in Adaptive and Personalized Systems (SOAP 2016)*. ACM.

[8] Tamas Jambor and Jun Wang. 2010. Optimizing Multiple Objectives in Collaborative Filtering. In *Proceedings of the Fourth ACM Conference on Recommender Systems (RecSys '10)*. ACM, New York, NY, USA, 55–62.

[9] Yuanchun Jiang and Yezheng Liu. 2012. Optimization of online promotion: a profit-maximizing model integrating price discount and product recommendation. *International Journal of Information Technology & Decision Making* 11, 05 (2012), 961–982.

[10] Paolo Massa and Paolo Avesani. 2007. Trust-aware recommender systems. In *Proceedings of the 2007 ACM conference on Recommender systems*. ACM, 17–24.

[11] Phong Nguyen, John Dines, and Jan Krasnodebski. 2017. A Multi-Objective Learning to re-Rank Approach to Optimize Online Marketplaces for Multiple Stakeholders. *arXiv preprint arXiv:1708.00651* (2017).

[12] Rodrygo LT Santos, Craig Macdonald, and Iadh Ounis. 2010. Exploiting query reformulations for web search result diversification. In *Proceedings of the 19th international conference on World wide web*. ACM, 881–890.

[13] Alexa Megan Sharp. 2007. *Incremental algorithms: solving problems in a changing world*. Cornell University.

[14] Hongzhi Yin, Bin Cui, Jing Li, Junjie Yao, and Chen Chen. 2012. Challenging the long tail recommendation. *Proceedings of the VLDB Endowment* 5, 9 (2012), 896–907.

[15] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. 2005. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*. ACM, 22–32.