Modeling Difficulty in Recommender Systems

Benjamin Kille, Sahin Albayrak
DAI-Lab
Technische Universität Berlin
{kille,sahin}@dai-lab.de

ABSTRACT

Recommender systems have frequently been evaluated with respect to their average performance for all users. However, optimizing such recommender systems regarding those evaluation measures might provide worse results for a subset of users. Defining a difficulty measure allows us to evaluate and optimize recommender systems in a personalized fashion. We introduce an experimental setup to evaluate the eligibility of such a difficulty score. We formulate the hypothesis that provided a difficulty score recommender systems can be optimized regarding costs and performance simultaneously.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval - Information filtering, Retrieval models, Search process, Selection process; H.3.4 [**Information Technology and Systems Applications**]: Decision support

General Terms

Algorithms, Design, Experimentation, Measurement, Human factors

Keywords

difficulty, recommender systems, user modeling, evaluation

1. INTRODUCTION

Evaluating a recommender system's performance represents a non-trivial task. The choice of evaluation measure and methodology depends on various factors. Modeling the recommendation task as rating prediction problem favors measures such as root mean squared error (RMSE) and mean absolute error (MAE) [7]. In contrast, mean average precision (MAP) and normalized discounted cumulative gain (NDCG) qualify as evaluation measures for an item ranking scenario [12]. In either case, two recommendation algorithms are compared with respect to their average performance on the full set of users. This entails that all users are treated equally. However, we argue that the difficulty of recommending items to users varies. Suppose we consider a recommender system with two users, Alice and Bob. Alice has rated a large number of items. Bob has recently started using the system and rated a few items. The system recommends a number of items to both of them. Alice and Bob rate the recommended items. Suppose we attempt to evaluate two recommender algorithms based on those ratings, denoted as R_1 and R_2. Assume that R_1 predicts Alice's ratings with an error of 0.8 and Bob's ratings with an error of 0.9. On the other hand, we observe R_2 to deviate 1.0 for Alice and 0.8 for Bob, respectively. Averaging the errors, we obtain 0.85 for R_1 and 0.9 for R_2. Still, R_2 predicts Bob's ratings better even though his preferences exhibit higher sparsity compared to Alice's. Besides the number of ratings, there are further factors discriminating how well an recommendation algorithm perform for a given user. We introduce the notion of difficulty in the context of recommender systems. Each user is assigned a difficulty value reflecting her expected evaluation outcome. Users with high errors or disordered item rankings receive a high difficulty value. Contrarily, we assign low difficulty values to users exhibiting low errors and well ordered item rankings. Recommender systems benefit from those difficulty value twofold. First, optimization can target difficult users who require such efforts. On the other hand, the recommender system can provide users with low difficulty values with recommendations generated by more trivial methods. Recommending most popular items represents such an approach. Second, difficulty values enable the system to estimate how likely a specific user will perceive the recommendations as adequate. Thereby, the system can control interactions with users, e.g. by asking for additional ratings to provide better recommendations for particularly difficult users.

2. RELATED WORK

Adomavicius and Tuzhilin reveal possible extensions to recommender systems [1]. They mention an enhanced understanding of the user as one such extension. Our work attempts to contribute to this by defining a user's difficulty. Hereby, the system estimates a user's likely perception of the recommendations. The methods performing best on the well-known *Netflix Prize Challenge* had applied ensemble techniques to further improve their rating prediction accuracy [3, 11]. Both do not state an explicit modeling of recommendation difficulty on the user-level. However, ensem-

Workshop on Recommendation Utitlity Evaluation: Beyond RMSE (RUE 2012), held in conjunction with ACM RecSys 2012 September 9, 2012, Dublin, Ireland.

ble techniques involve an implicit presence of such a concept. Combining the outcome of several recommendation algorithms does make sense in case a single recommendation algorithm fails for a subset of users. Such an effect is consequently compensated by including other recommendation algorithms' outcomes. Bellogin presents an approach to predict a recommender systems performance [4]. However, the evaluation is reported on the system level averaging evaluation measures over the full set of users. Our approach focuses on predicting the user-level difficulty. Koren and Sill introduced a recommendation algorithms that outputs confidence values [8]. Those confidence values could be regarded as difficulty. However, the confidence values are algorithm specific. We require the difficulty score's validity to generally hold. The concept of evaluation considering difficulty has been introduced in other domains. Aslam and Pavlu investigated estimation of a query's difficulty in the context of information retrieval [2]. Their approach bases on the diversity of retrieved lists of documents by several IR systems. Strong agreement with respect to the document ranking indicates a low level of difficulty. In contrast, highly diverse rankings suggest a high level of difficulty. He et al. describe another approach to estimate a query's difficulty [6]. Their method compares the content of retrieved documents and computes a coherence score. They assume that in case highly coherent documents are retrieved the query is clear and thus less difficult than a query resulting in minor coherency. Genetic Algorithms represent another domain where difficulty has received the research community's attention. However, the difficulty is determined by the fitness landscape of the respective problem [5, 6]. Kuncheva and Whitaker investigated diversity in the context of a classification scenario [9]. Concerning a query's difficulty, diversity has been found an appropriate measure [2]. We will adapt two diversity measures applied in the classification scenario for determining the difficulty to recommend a user items.

3. METHOD

We strive to determine the difficulty of the recommendation task for a given user. Formally, we are looking for a map $\delta(u) : \mathcal{U} \mapsto [0; 1]$ that assigns each user $u \in \mathcal{U}$ a real number between 0 and 1 corresponding to the level of difficulty. For one recommendation algorithm the difficulty for a given user could be simply determined by a (normalized) evaluation measure, e.g. RMSE. However, such a difficulty would not be valid for other recommendation algorithms. In addition, recommender systems optimized with respect to the item ranking would likely end up with different difficulty values. We adapt the idea of measuring difficulty in terms of diversity to overcome those issues. We assume the recommender systems comprises several recommendations methods, denoted $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$. Each such A_n generates rating predictions along with item rankings for a given user u. We choose RMSE to evaluate the rating predictions and NDCG to assess the item ranking, respectively. We measure a user's difficulty by means of the diversity of those rating predictions and item rankings. For that purpose, we adjusted two diversity metrics introduced by Kuncheva and Whitaker for classification ensembles [9]. We alter the pair-wise Q statistics to fit the item ranking scenario. Regarding the rating prediction we adapt the difficulty measure θ.

3.1 Q statistics

Applied to the classification ensemble setting, the Q statistics base on a confusion matrix. The confusion matrix confronts two classifiers in a binary manner - correctly classified versus incorrectly classified. We need to adjust this setting to fit the item ranking scenario. Table 1 illustrates the adjusted confusion matrix. We count all correctly ranked items as well as all incorrectly ranked items for both recommendations algorithms. Subsequently, the confusion matrix displays the overlap of those items sets. Equation 1 represents the Q statistic derived from the confusion matrix. Note that the Q statistic measures diversity in between two recommender algorithms. Hence, we need to average the Q statistic values obtained for all comparisons in between the available algorithms. Equation 2 computes the final diversity measure.

$$Q_{ij}(u) = \frac{N^{11}N^{00} - N^{01}N^{10}}{N^{11}N^{00} + N^{10}N^{01}} \tag{1}$$

$$Q_{\mathcal{A}}(u) = \binom{|\mathcal{A}|}{2}^{-1} \sum_{A_i, A_j \in \mathcal{A}} Q_{ij} \tag{2}$$

3.2 Difficutly measure θ

Kuncheva and Whitaker introduce the difficulty measure θ as a non-pairwise measure [9]. θ allows us to consider several recommendation algorithms simultaneously. We iterate over the set of withhold items for the given user. At each step we compute the RMSE for all recommendation algorithms at hand. Thereby, we observe the variance in between all recommender algorithms' RMSE values. The average variance displays how diverse the user is perceived by the set of recommendation algorithms. Equation 3 calculates θ. $\mathcal{I}$ denotes the set of items in the target user's test set. The $\sigma(i)$ function computes the variances in between the recommendation algorithms' RMSE values for the given item.

$$\theta(u) = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \sigma(i) \tag{3}$$

3.3 Difficulty measure δ

We attempt to formulate a robust difficulty measure for recommender systems. Therefore, we propose a combination of the Q statistic and θ. As a result, both RMSE and NDCG are considered. The more recommendation algorithms we include the more robust the final difficulty value will be. Equation 4 displays a linear combination of both measures. The parameter $\lambda \in [0; 1]$ controls the weighting. $\lambda = 1$ allows to focus on the ranking task. The absence of rating values might require such a setting.

$$\delta(u|\mathcal{A}) = \lambda \cdot Q_{\mathcal{A}}(u) + (1 - \lambda) \cdot \theta(u) \tag{4}$$

4. EXPERIMENTAL SETTING

We will evaluate the proposed difficulty measure δ and subsequently outline the intended experimental protocol. Several data sets, such as *Movielens 10M, Netflix* and *Last.fm* provide the required data. We will implement item-based

	A_i: rank(k) $\leq$ rank(l)	A_i: rank(k) $>$ rank(l)
A_j: rank(k) $\leq$ rank(l)	N^{11}	N^{10}
A_j: rank(k) $>$ rank(l)	N^{01}	N^{00}

Where rank(k) $\leq$ rank(l) is true.

Table 1: Adjusted confusion matrix

and user-based neighborhood recommenders, matrix factorization recommenders along with slope-one recommenders. All those recommendation methods do not require data except user preferences. As a first step, we split each data sets in three partitions. The first partition will be used for training and the second partition to assign each user a difficulty score according to Equation 4. Finally, the third partition will be used as evaluation data set. The subset of users with comparably low difficulty score will receive recommendations based on a non-optimized recommendation method such as the most popular recommender. The subset of users with medium difficulty score will receive recommendations generated by slightly optimized recommender algorithms. Finally, the particularly hard users will receive recommendations generated by highly optimized recommenders. We will compare both required efforts and recommendation quality against approaches dealing with all users in the same ways. Such approaches will include optimizing recommenders to perform well averaged over the full set of users and recommending all users with trivial recommenders, for instance most popular recommendations. Our hypothesis is that the difficulty score will allow us to achieve lower costs along with a comparably high recommendation quality by selecting an appropriate recommendation algorithm for a given user. In addition, we will observe what user characteristics determine her difficulty.

5. CONCLUSION AND FUTURE WORK

We introduced the notion of difficulty in the context of recommending items to users. Measuring that task's difficulty on user-level allows a more personalized optimization of recommender systems. Users with comparably low difficulty scores receive adequate recommendations without much efforts. On the other hand, users with a comparably high difficulty score may be asked to provide additional data to improve the system's individual perception. A combination of the Q statistic and the difficulty measure θ - both adjusted to fit the recommendation scenario - allows us to measure the difficulty for a given user. The calculation requires a sufficiently large data set containing user preferences on items along with a set of implemented recommendation algorithms. We introduced an experimental setting that we will use to evaluate the presented methodology in section 4. In addition, we intend to apply pattern recognition techniques to find correlations between the difficulty score and user characteristics. For instance, the number of ratings is known to correlate with the difficulty for new users ("cold-start problem") and users with a large number of items ("power-users") [10].

6. ACKNOWLEDGEMENTS

This work is funded by the DFG (Deutsche Forschungsgemeinschaft) in the scope of the LSR project.

7. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transaction on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[2] J. A. Aslam and V. Pavlu. Query hardness estimation using jensen-shannon divergence among multiple scoring functions. In *Proceedings of the 29th European conference on IR research*, ECIR'07, pages 198–209, 2007.

[3] R. M. Bell and Y. Koren. Lessons from the netflix prize challenge. *SIGKDD Explor. Newsl.*, 9(2):75–79, 2007.

[4] A. Bellogin. Predicting performance in recommender systems. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 371–374. ACM, 2011.

[5] M. Hauschild and M. Pelikan. Advanced neighborhoods and problem difficulty measures. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, GECCO '11, pages 625–632, 2011.

[6] J. He, M. Larson, and M. De Rijke. Using coherence-based measures to predict query difficulty. In *Proceedings of the IR research, 30th European conference on Advances in information retrieval*, ECIR'08, pages 689–694, 2008.

[7] J. Herlocker, J. Konstan, L. Terveen, and J. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, 2004.

[8] Y. Koren and J. Sill. Ordrec: an ordinal model for predicting personalized item rating distributions. In *Proceedings of the fifth ACM conference on Recommender systems*, RecSys '11, pages 117–124, New York, NY, USA, 2011. ACM.

[9] L. Kuncheva and C. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51:181–207, 2003.

[10] A. Said, B. Jain, and S. Albayrak. Analyzing weighting schemes in collaborative filtering: Cold start, post cold start and power users. In *27th ACM Symposium On Applied Computing (SAC '12)*, New York, NY, USA, 2012. ACM.

[11] G. Takacs, I. Pilaszy, B. Nemeth, and D. Tikk. Major components of the gravity recommendation system. *SIGKDD Explorations*, 9(2), 2007.

[12] S. Vargas and P. Castells. Rank and relevance in novelty and diversity metrics for recommender systems. In *Proceedings of the fifth ACM conference on Recommender systems*, RecSys '11, pages 109–116, New York, NY, USA, 2011. ACM.