User Partitioning Hybrid for Tag Recommendation

Jonathan Gemmell, Bamshad Mobasher, and Robin Burke

Center for Web Intelligence
School of Computing, DePaul University
Chicago, Illinois, USA
{jgemmell,mobasher,rburke}@cdm.depaul.edu

Abstract. Tag recommendation is a fundamental service in today's social annotation systems, assisting users as they collect and annotate resources. Our previous work has demonstrated the strengths of a linear weighted hybrid, which weights and combines the results of simple components into a final recommendation. However, these previous efforts treated each user the same. In this work, we extend our approach by automatically discovering partitions of users. The user partitioning hybrid learns a different set of weights for these user partitions. Our rigorous experimental results show a marked improvement. Moreover, analysis of the partitions within a dataset offers interesting insights into how users interact with social annotations systems.

Keywords: User Partitioning, Personalization, Social Annotation Systems, Tag Recommendation, Hybrid Recommendation.

1 Introduction

Social annotation systems allow users to collect online resources such as music, videos, products or journal articles. The defining characteristic of these systems is the ability of users to assign tags to resources. The complex interaction of these users, resources and tags form a large multidimensional information space, often called a folksonomy. While these information spaces connect users to content in interesting and complex ways, they are also inherently difficult to navigate and benefit from recommender technologies.

Many forms of recommendation are possible in social annotation systems. Resources, tags or even users are all potential outputs for a recommender. In many cases, additional constraints can be applied. For example, a user may desire a resource recommendation based upon a tag they have clicked on or a resource they have recently consumed. In tag recommendation, the subject of this paper, the system recommends a set of tags to a user as she is annotating a resource. This recommendation reduces user effort, helps the user avoid misspellings or errors, and maintains a cleaner tag space.

Our previous work in tag recommendation for social annotation systems has focused on the use of linear weighted hybrids that relies on the combined efforts of several simple component recommenders. A key step is the generation of weights for each component, indicating how much influence a component will have in the recommendation. Experimental work has shown that different folksonomies require different sets of weights to produce optimal results.

V. Dimitrova et al. (Eds.): UMAP 2014, LNCS 8538, pp. 74–85, 2014.

This observation has led us to infer that users employ these systems in different ways. For example, in a social annotation system where users annotate journal articles, approaches that model users or resources as tags do well and get higher weights. It seems that scientists are careful about the tags they select and often draw tags from their domain of expertise. In contrast, models representing users as a vector of tags tend to do poorly in a system where users annotate music. These users appear more likely to use generic tags like "rock" and idiosyncratic tags such as "sawAtConcertWithSuzy." Yet, these inferences appear overly broad. Not every scientist is careful about the use of tags and not every music fan is careless.

In this work, we partition users and learn a separate set of weights for each partition. One approach may be to partition users based on some metric such as the size of the user profile or the variability of the user's resources. However, a preconceived metric may not produce optimal results. Instead, we attempt to automatically discover the partitions using an approach similar to the one in K-means clustering.

We first assign users to a random partition. Weights are trained for each partition, and users are then reassigned to the partition in which they best perform. This procedure is repeated until the system reaches equilibrium.

Our experimental findings show that a user partitioning hybrid outperforms previous approaches. The technique has the added benefits of being efficient, scalable and extensible. Finally, the partitions and their trained weights offer insights into how different segments of users interact with social annotation systems.

The rest of this paper is organized as follows. In Section 2 we discuss related work. Section 3 describes our data model and how we partition users for the user partitioning hybrid. Our experimental evaluation is presented in Section 4. Section 5 offers conclusions.

2 Related Work

A common thread found in research focusing on tag recommendation is the need for an integrative approach. FolkRank [9] is a variant of the well-known PageRank algorithm. The graph it induces from the folksonomy connects users, resources and tags. While known to be accurate, it suffers from the fact that it is computationally expensive to run.

Tucker decomposition has been used to factor the three dimensional tagging data into three feature spaces and a core residual tensor [18]. Once computed the matrices can be used to quickly recommend tags. However, the calculation of the matrices is prohibitive.

Pair-wise interaction tensor factorization [14, 15] formed the basis for the winning submission of the PKDD 2009 Tag Recommendation Challenge [12]. It is a model-based approach that generates factor matrices using a set of positive and negative examples. An iterative gradient-descent algorithm is used to optimize a ranking function that prefers positive examples over negative ones. We include $PITF$ in our experimental work as a means of comparison.

Our previous work in tag recommendation has demonstrated the benefits of hybrid recommenders [3–5]. In this paper, we extend those efforts.

3 Tag Recommendation Based on User Partitioning

This section first describes the data model used to represent social annotation systems. Our user partitioning hybrid is then presented. So too are the the component recommenders that make up the hybrid.

3.1 Data Model

A social annotation system can be modeled with four sets. The first set, *U*, includes the set of users that employ the system. The second set, *R*, includes the all resources that any of the users have collected. The third set, *T*, includes every tag any user has ever applied to a resource. The final set, *A*, is the set of annotations. An annotation includes a single user, a single resource, and all tags that user has applied to that resource.

This model can be translated to a three-dimensional matrix, which we call *URT*. An entry *URT(u,r,t)* is 1 if *u* annotated *r* with *t*, and is otherwise 0. This three-dimensional matrix offers the convenience of generating aggregate projections. Aggregate projections reduce the dimensionality of the data making it easier to work with but sacrifices some information [6, 13].

One such projection is *RT*. In this projection, an entry *RT(r,t)* is calculated as the number of users that have assigned *t* to *r*. Two other aggregate projections are possible, *UR* and *UT*. In these projections, an entry can be calculated as the number of times *u* has tagged a resource *r* or the number of times he has employed a tag *t* respectively. Given these projection we are able to model a user by drawing a row from either *UR* or *UT*. In the first case, the user is being modeled as a vector over the resource space, where the weight $w(r_i)$ in dimension *i* corresponds to the importance of resource, r_i: $\boldsymbol{u^r} = \langle w(r_1), w(r_2)...w(r_{|R|})\rangle$.

Likewise, a user could be modeled over the tag space to produce $\boldsymbol{u^t}$. Analogous models can be created for resources ($\boldsymbol{r^u}$,$\boldsymbol{r^t}$) and tags ($\boldsymbol{t^u}$,$\boldsymbol{t^r}$) by drawing either a row or a column from one the projections. Previous experimentation has shown that a binary version of *UR* yields better results and we continue to use the binary version in this work.

3.2 User Partitioning Hybrid

The motivation behind our approach to automatic user partitioning stems from the observation that users exhibit complex patterns in how they interact with social annotation systems. For example, some users may consistently employ popular tags, while others prefer idiosyncratic tags. If we could identify or predict a user's behavior, we might be able to select a recommendation strategy that best suits that behavior.

One possible approach would be to partition users based on some preconceived notion and then independently design a recommendation strategy for each partition. However, there is no guarantee that these partitions would be optimal or that the partitioning strategy would generalize across all social annotation systems. In this work, we instead opt to automatically identify optimal partitions through an iterative process similar to K-means clustering [7].

First, users are randomly placed into one of k partitions. A recommendation strategy is optimized for each partition of users. Then using a holdout set, each user is evaluated against the optimized strategies. User are reassigned to the partition in which they perform best. The recommenders are once again optimized against their new collection of users. This process iteratively repeats until the partitions stabilize.

While the approach is similar to K-means clustering, notice that users are not assigned to partitions based on a similarity metric between themselves or to a partition-mean. Users are instead assigned to a partition based on their performance in the partition. In this manner, users are partitioned based on their affinity to an optimized recommendation strategy.

More specifically, we begin by evaluating the function $\psi : U \times R \times T \rightarrow \mathbb{R}$ where a user $u \in U$, a resource $r \in R$ and a tag $t \in T$ results in a real-valued result p. This value represents the prediction of how well the tag is suited for that particular user-resource pair: $\psi(u, r, t) = p$. In order to generate a recommendation list, a ranked list of suggested tags for a particular user and resource is generated. Given a user u and resource r, we iterate over all tags and calculate their relevance. Finally, we sort the tag by this results and return the top n tags: $rec(u, r) = TOP^n_{t \in T} \psi(u, r, t)$.

Our user partitioning hybrid is built from simple components that focus on specific dimensions of the data [1]. The results are then aggregated to form the final recommendation. If each component is able to generate its own score for a user-resource-tag triple, then the hybrid can combine these scores.

More specifically, the hybrid takes a collection of tag recommenders C. When asked to make a recommendation for a user u and resource r, it will iterate over all tags querying each of its component recommenders, $c \in C$, for a tag, t, and combine the results in the linear model: $\psi_h(u, r, t) = \sum_{c \in C} \alpha_c \psi_c(u, r, t)$ where $\psi_h(u, r, t)$ is the linear weighted relevance score of the tag and α_c is the weight given to the component, c. Scores from the components are not guaranteed to be on the same scale. We therefore normalize the scores so that each $\psi_c(u, r, t)$ falls in the interval [0,1].

The linear weighted hybrid can include any number of components. In order to maximize its accuracy, we train their weights using a hill climbing technique. At first, the α vector is initialized with random positive numbers such that the sum of the vector is 1. A holdout set is used to evaluate the performance of the hybrid with those weights. We rely on recall (see Section 4.2).

The α vector is then randomly modified and tested against the holdout set again. If the performance is improved, we keep the change; otherwise, we reject it. Two small modifications ensure we are not trapped in a local maxima. First, we occasionally accept a change to the α vector even when it does not improve the performance. Second, we randomly restart, so that we may thoroughly examine the α space.

The α vectors in our previous work on linear weighted hybrids were trained against all users in the dataset. As a result, recommendations for users were drawn from the same combination of component recommenders. Our user partitioning hybrid segments the users and trains a unique α vector for each partition of users. After optimizing the α vectors for each partition, we reassign the users. Using the same holdout set, we calculate a user's recall for each partition and reassign the user to the partition in which

he performs best. The optimization-reassignment procedure is repeated until the users no longer move from partition to partition.

3.3 Component Recommenders

Any number of components can be included in the user partitioning hybrid. We have purposely chosen simple components that exploit only a few dimensions of the data. Our goal is to create an integrative recommender by combining these components rather than building a single complex recommender. We now describe those components.

Popularity Models. Given the user-resource pair, the component may recommend the most popular tags for that particular resource. This strategy ignores the user and is strictly resource dependent. We define $\psi(u, r, t)$ for the resource based popularity recommender, pop_r, as $\psi(u, r, t) = \sum_{v \in U} \theta(v, r, t)$ where $\theta(v, r, t)$ is 1 if v tagged r with t and 0 otherwise. Likewise, a component may merely recommend the most popular tags for a particular user. This approach does not consider the resource and consequently may recommend some irrelevant tags. We define $\psi(u, r, t)$ for the user based popularity recommender, pop_u, as $\psi(u, r, t) = \sum_{s \in R} \theta(u, s, t)$.

User-Based Collaborative Modeling. Collaborative filtering works under the premise that patterns exist in how users interact with the system. User-based collaborative filtering [8, 11, 17] looks for similarities among users. For our work in tag recommendation, a neighborhood, N_r, of the k most similar users to u is identified through a similarity metric such that all the neighbors have tagged r. For any given resource the weighted sum can then be calculated as $\psi(u, r, t) = \sum_{v \in N_r} \sigma(u, v)\theta(v, r, t)$ where $\sigma(u, v)$ is the similarity between the users u and v.

Previous experiments have looked at many types of similarity metrics; we have found cosine similarity to most consistently produce the best results. When users are modeled as resources we call this approach KNN_{ur}. When users are modeled as tags we call this technique KNN_{ut}.

Item-Based Collaborative Modeling. Another form of collaborative filtering is item-based collaborative filtering [2, 16]. Rather than finding similarities among users, this approach creates a neighborhood of similar resources. As before, we notice that there are two ways in which we might model resources.

The model KNN_{ru} treats resources as a vector over the user space. The model KNN_{rt} treats resources as a vector over the tag space. We define N_u as the k nearest resources to r drawn from the user profile, u, and then define the relevance score of a tag for a user-resource pair as $\psi(u, r, t) = \sum_{s \in N_u} \sigma(r, s)\theta(u, s, t)$.

4 Experimental Evaluation

In this section, we provide information about our three real world datasets, including how we collected and preprocessed the data. We then describe our experimental methodology. We first limit our analysis to the individual datasets and then draw larger conclusions.

4.1 Datasets

Our exhaustive experimental analysis was conducted on three data sets: Citeulike, MovieLens and LastFM. After collecting the data, we generated p-cores [10]. A subset of the data was selected such that each remaining user, resource and tag is guaranteed to occur in at least p annotations, where an annotation represents a user, a resource and all tags applied by that user to that resource.

Extracting p-cores from the data discards information – it eliminates infrequent users, resources and tags – but it also yields some benefits. It reduces noise and enables algorithms than might otherwise be computationally expensive. Infrequent items are necessary when experimenting on the "long tail" or when investigating the cold start problem. This paper, however, is focused on the complex interactions between users, resources and tags. To this end, we are focused on the denser part of the graph.

Citeulike is used to manage and organize journal articles. It is mainly used by scientists and researchers. Data is available to download directly from the site. In this paper, we use a snapshot taken from 17 February 2009. While newer datasets are available, we opted to use the same dataset as in our previous work in order to maintain consistency. After we computed a 5-core, the dataset contained 2,051 users, 5,376 resources, 3,343 tags and a total of 105,873 annotations.

MovieLens is a movie recommendation website run by the GroupLens research lab at the University of Minnesota. They collect several datasets from their users and make these datasets available to researchers. In particular, one of those datasets contains tagging information. We created a 5-core from this data. The result was 35,366 annotations with 819 users, 2,445 resources and 2,309 tags.

LastFM provides many services for their users. User can upload playlists, share their tastes and connect to fans with similar interests. It also allows users to annotate songs, albums and artists. After selected 100 random users, and recursively crawling the "friend" network we were able to download user profiles for thousands of users. This data was denser than the previous datasets and permitted a p-core of 20. It contains 2,368 users, 2,350 resources, 1,141 tags and 172,177 annotations. These experiments focus on the album data, but parallel experiments show similar trends on the artist and song datasets.

4.2 Methodology

In order to train and evaluate our techniques we divide each of the user's annotations among five folds. We use four of these folds to build our recommenders. The fifth is used as training data. We use this fifth tune the model parameters; for example when selecting the number of neighbors for our collaborative filtering components. Moreover, we use this fifth fold during the training of the component weights and the reassignment of users to new partitions. We then discard the fifth folded, performing four fold cross validation on the remaining folds with the discovered parameters.

Given a testing annotation, we can submit the user u and resource r to a recommendation engine to produce set of recommended tags, T_r. Comparing these tags to the holdout set, T_h, we can evaluate our recommender with on recall and precision.

Recall is a common metric for evaluating the utility of recommendation algorithms. It measures the percentage of items in the holdout set that appear in the recommendation set. Recall is a measure of completeness and is defined as $recall = |T_h \cap T_r|/|T_h|$.

Precision is another common metric for measuring the usefulness of recommendation algorithms. It measures the percentage of items in the recommendation set that appear in the holdout set. Precision measures the exactness of the recommendation algorithm and is defined as $precision = |T_r \cap T_r|/|T_r|$.

The recall and precision will vary depending on the size on the recommendation set. In the following experiments we present the metrics with recommendation sets of size one through ten.

4.3 Experimental Results

Table 1 shows relative contribution of the component recommenders as learned through the hill climbing approach for Citeulike, MovieLens and LastFM. The first line, labeled "all", describes the contributions when all users are used to train the α vectors. In this approach, there are no partitions. For example, we see that the linear weighted hybrid has little use for Pop_u or Pop_r in Citeulike, but instead relies mostly on KNN_{rt} (50.9%). There are 2051 users in total.

The next five lines, labeled "1" through "5", describe the contribution of the component recommenders for five partitions. In this case, the α vectors were trained only on the users in the partitions. As described above, users were then reassigned to the partition in which they best performed. This process repeated until the partitions stabilized. We can see that partition 1 contains 264 users and that these users rely more strongly on Pop_u and Pop_r, 19.1% and 16.7% respectively.

Figures 1 through 3 present the performance of the algorithms. Recommendation sets were generated of size one through ten. These sets were then evaluated in terms of recall and precision. Each line in the graphs represents a particular recommendation technique. For example, we see that the worst performing approach in Citeulike is KNN_{ur}.

Citeulike. In Citeulike, we see that the three leading approaches are the interaction technique ($PITF$), the linear weighted hybrid ($Hybrid$) and the user partitioning hybrid (UPH). These results confirm our supposition that an integrative approach is needed which draws upon multiple dimensions of the data. When looking at their recall for ten tags, they all achieve nearly identical results.

In this dataset, it seems there is little advantage in partitioning users. The reason can be seen when inspecting the α vectors. Notice that when the hybrid is trained for all users, KNN_{rt} dominates the other components (50.9%). The next strongest contributor is KNN_{ut} (26.5%). In this domain, it seems that modeling resources and users in the tag space is to be preferred. This is not surprising since the users of Citeulike are researchers that are usually careful about how they apply their tags, often drawing on keywords from their area of expertise.

If we turn our attention to the user partitioning hybrid, we see that four of the five partitions have α vectors quite similar to that of the linear weighted hybrid. Partitions

Table 1. Contributions of the component recommenders to the the linear weighted hybrid and the user partitioning hybrid

Citeulike

	n	Pop_u	Pop_r	KNN_{ur}	KNN_{ut}	KNN_{ru}	KNN_{rt}
All	2051	0.007	0.034	0.066	0.265	0.109	0.509
1	264	0.191	0.167	0.049	0.208	0.253	0.133
2	591	0.017	0.043	0.058	0.173	0.110	0.600
3	304	0.099	0.036	0.143	0.081	0.206	0.435
4	541	0.026	0.101	0.044	0.366	0.029	0.433
5	351	0.098	0.061	0.009	0.098	0.155	0.602

MovieLens

	n	Pop_u	Pop_r	KNN_{ur}	KNN_{ut}	KNN_{ru}	KNN_{rt}
All	819	0.028	0.023	0.063	0.407	0.048	0.431
1	241	0.126	0.028	0.067	0.177	0.185	0.418
2	142	0.040	0.083	0.063	0.436	0.106	0.273
3	120	0.162	0.106	0.137	0.156	0.214	0.226
4	94	0.153	0.177	0.208	0.082	0.152	0.228
5	222	0.053	0.086	0.008	0.160	0.101	0.597

LastFM

	n	Pop_u	Pop_r	KNN_{ur}	KNN_{ut}	KNN_{ru}	KNN_{rt}
All	2368	0.017	0.032	0.011	0.471	0.430	0.038
1	347	0.187	0.083	0.383	0.304	0.216	0.173
2	465	0.036	0.035	0.125	0.039	0.403	0.362
3	401	0.179	0.205	0.019	0.340	0.162	0.094
4	750	0.011	0.047	0.024	0.425	0.383	0.110
5	405	0.061	0.158	0.060	0.340	0.335	0.047

2 through 4 all largely rely on KNN_{rt} and KNN_{ut}. It is only the first partition – and the smallest – that appears to require a uniquely different combination of components, relying more strongly on the popularity based techniques than any of the other partitions. These observations suggest not only that modeling users and resource as a vector of tags is preferred, but also that users are relatively uniform in how they interact with the system. The result is that a partitioning of users, offers little additional benefit.

MovieLens. Figure 2 show the performance of the recommendation techniques on the MovieLens dataset. Again, $PITF$, $Hybrid$ and UPH lead the pack. In this case, $Hybrid$ lags behind and UPH improves upon it enough to match $PITF$. Again, examination of the α vectors offer insights.

As in before, the linear weighted hybrid trained on all 819 MovieLens users is largely influenced by KNN_{rt} and KNN_{ut}. It seems that users are annotating movies with keywords drawn from the domain such as “drama” or using the names of actors. However,

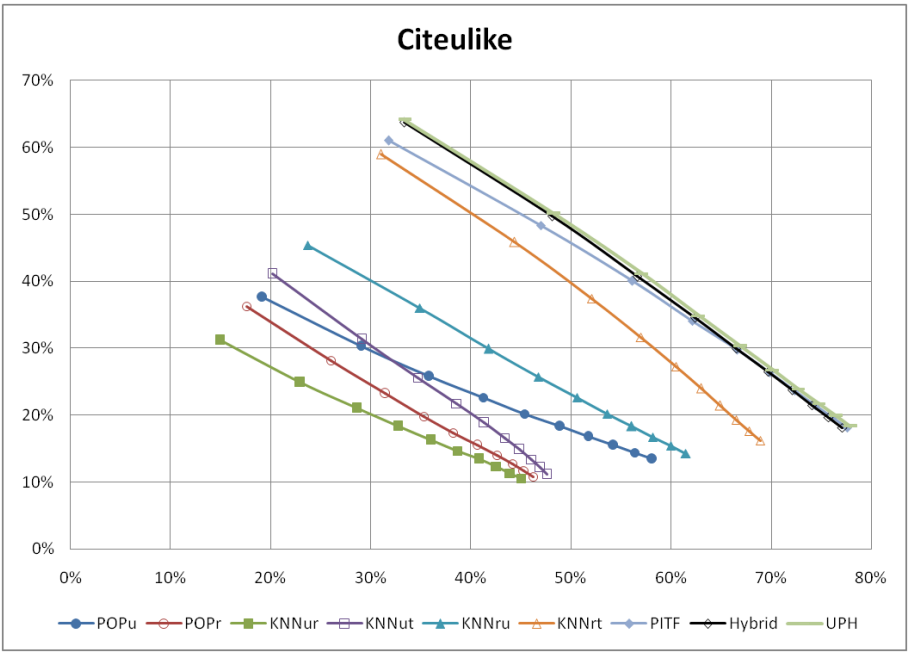

Fig. 1. Citeulike: recall versus precision

unlike Citeulike, when we inspect the α vectors for the user partitioning hybrid we see more variation.

Users in the first partition appear be influenced by KNN_{rt} and KNN_{ut}, but also tend reuse tags as evidenced by the weights in Pop_u and KNN_{ru}. Users in the third and fourth partitions rely on a combination of all the component recommenders. It is the second and fifth partitions that are most similar to the linear weighted hybrid. These two partitions account the plurality of the users but not to the degree we saw above. Consequently, there is a greater benefit for automatically partitioning users and training the component weights for each one. In short, it seems that there is more variance in how these users are interacting with the social annotation system.

LastFM. In our final dataset, we see that UPH outperforms $PITF$ and $Hybrid$. It would seem that users annotating resources in LastFM exhibit a greater variety of patterns than we observed above. The linear weighted hybrid is composed largely of KNN_{ut} and KNN_{ru}. The other components contribute very little.

It is the fourth partition of the user partitioning hybrid that most closely resembles the linear weighted hybrid. We see that 750 users were assigned to this partition, the largest allocation by far. This large crowd of users was likely influencing the linear weighted hybrid, which trains the model on all users.

However, when users are separated into partitions, these users consolidate around other combinations of component recommenders. We see users in the first partition

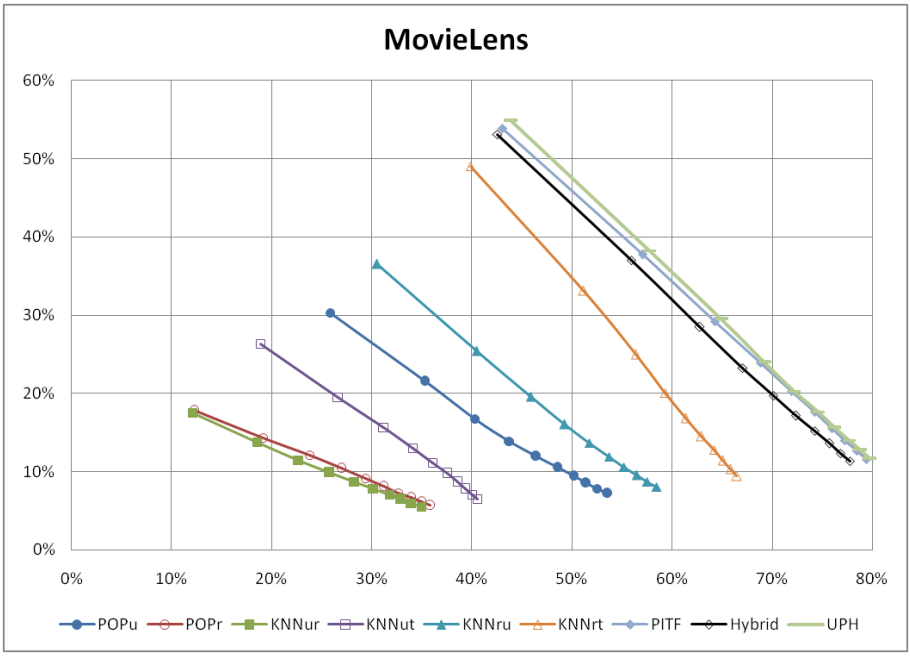

Fig. 2. MovieLens: recall versus precision

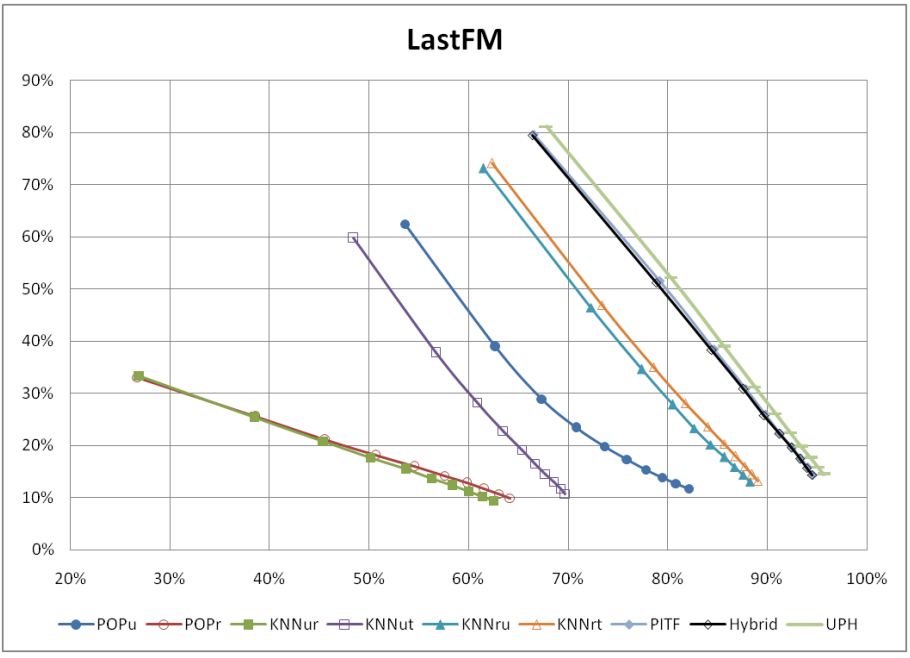

Fig. 3. LastFM: recall versus precision

taking advantage of KNN_{ur}, users in the second relying KNN_{rt}, users in the third partition benefiting from almost all the components except KNN_{ur}, and users in partition five putting more emphasis on Pop_r. This variation among the α vectors and the corresponding improvement of the user partitioning hybrid over the linear weighted hybrid support the notion that automatic user partitioning can improve recommenders by catering to the needs of groups of users rather than trying to build a single model to satisfy them all.

5 Conclusion

In this paper, we introduced a framework for tag recommendation based on automatically partitioning users. A user partitioning hybrid was constructed from several simple component recommenders. A vector of weights controlled the contribution of the components to the hybrid. When users are partitioned, a vector is trained for each partition. Users were then reassigned to the partition in which they performed best. The process repeated until the partitions stabilized. Our experimental evaluation on three real world datasets shows that automatic user partitioning can improve performance of a linear weighted hybrid. Moreover, examination of the components and their contribution can lead to valuable insights with regard to how users are interacting with different social annotation systems. Our future work aims to generalize this approach and apply it to other recommendation tasks in social annotation systems such as resource recommendation.

References

1. Burke, R.: Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction 12(4), 331–370 (2002)
2. Deshpande, M., Karypis, G.: Item-Based Top-N Recommendation Algorithms. ACM Transactions on Information Systems 22(1), 143–177 (2004)
3. Gemmell, J., Ramezani, M., Schimoler, T., Christiansen, L., Mobasher, B.: A fast effective multi-channeled tag recommender. In: European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases Discovery Challenge, Bled, Slovenia (2009)
4. Gemmell, J., Schimoler, T., Mobasher, B., Burke, R.: Improving folkrank with item-based collaborative filtering. In: Recommender Systems & the Social Web, New York (2009)
5. Gemmell, J., Schimoler, T., Mobasher, B., Burke, R.: Hybrid tag recommendation for social annotation systems. In: 19th ACM International Conference on Information and Knowledge Management, Toronto, Canada (2010)
6. Gemmell, J., Schimoler, T., Ramezani, M., Mobasher, B.: Adapting K-Nearest Neighbor for Tag Recommendation in Folksonomies. In: 7th Workshop on Intelligent Techniques for Web Personalization and Recommender Systems, Chicago, Illinois (2009)
7. Hartigan, J.A., Wong, M.A.: Algorithm as 136: A k-means clustering algorithm. Journal of the Royal Statistical Society. Series C (Applied Statistics) 28(1), 100–108 (1979)
8. Herlocker, J., Konstan, J., Borchers, A., Riedl, J.: An Algorithmic Framework for Performing Collaborative Filtering. In: 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Berkeley, California. ACM (1999)

9. Hotho, A., Jäschke, R., Schmitz, C., Stumme, G.: Information Retrieval in Folksonomies: Search and ranking. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 411–426. Springer, Heidelberg (2006)
10. Jäschke, R., Marinho, L., Hotho, A., Schmidt-Thieme, L., Stumme, G.: Tag Recommendations in Folksonomies. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) PKDD 2007. LNCS (LNAI), vol. 4702, pp. 506–514. Springer, Heidelberg (2007)
11. Konstan, J., Miller, B., Maltz, D., Herlocker, J., Gordon, L., Riedl, J.: GroupLens: Applying Collaborative Filtering to Usenet News. Communications of the ACM 40(3), 87 (1997)
12. Marinho, L., Preisach, C., Schmidt-Thieme, L., Cantador, I., Vallet, D., Jose, J., Cao, H., Xie, M., Xue, L., Liu, C., et al.: ECML PKDD Discovery Challenge 2009-DC09 (2009)
13. Mika, P.: Ontologies are us: A unified model of social networks and semantics. Web Semantics: Science, Services and Agents on the World Wide Web 5(1), 5–15 (2007)
14. Rendle, S., Schmidt-Thieme, L.: Factor Models for Tag Recommendation in BibSonomy. In: ECML/PKDD 2008 Discovery Challenge Workshop, part of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, Bled, Slovenia (2009)
15. Rendle, S., Schmidt-Thieme, L.: Pairwise Interaction Tensor Factorization for Personalized Tag Recommendation. In: Proceedings of the Third ACM International Conference on Web Search and Data Mining, New York (2010)
16. Sarwar, B., Karypis, G., Konstan, J., Reidl, J.: Item-Based Collaborative Filtering Recommendation Algorithms. In: 10th International Conference on World Wide Web, Hong Kong, China (2001)
17. Shardanand, U., Maes, P.: Social Information Filtering: Algorithms for Automating "Word of Mouth". In: SIGCHI Conference on Human Factors in Computing Systems, Denver, Colorado (1995)
18. Symeonidis, P., Nanopoulos, A., Manolopoulos, Y.: Tag recommendations based on tensor dimensionality reduction. In: Proceedings of the 2008 ACM Conference on Recommender Systems, Lausanne, Switzerland (2008)