SAND2010-5291P

Evaluating graph based proximity search for a recommender system with item tags

Pooya Esfandiar
University of British Columbia
2366 Main Mall
Vancouver, BC
pooyae@cs.ubc.ca

Mohammad Khabbazhaye Tajer
University of British Columbia
2366 Main Mall
Vancouver, BC
mkhabbaz@cs.ubc.ca

David F. Gleich
Sandia National Labs
7011 East Ave. MS 9159
Livermore, CA
dfgleic@sandia.gov

Laks V.S. Lakshmanan
University of British Columbia
2366 Main Mall
Vancouver, BC
laks@cs.ubc.ca

ABSTRACT

Standard recommender systems use data from users and their rating behavior in order to recommend items of possible interest to them. Collaborative tagging systems like del.icio.us, IMDb, and flickr are becoming increasingly popular and users are finding them increasingly useful. Compared to conventional recommender systems, such tagging systems offer additional rich data in the form of users assigning tags to items. It is thus interesting to ask whether this additional information can be put to use while generating recommendations. We hypothesize that incorporating item based tags into a recommender system will improve its performance. We model such a system as a tri-partite graph of users, items and tags and use this graph to define a scoring function making use of graph-based proximity measures. Exactly calculating the item scores is computationally expensive and we use an efficient approximation algorithm to calculate scores. The usefulness and efficiency of the model are compared to a simple, non-graph based, approach. We evaluate these models on a combination of the Netflix ratings data and the IMDb tag data.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information Filtering*

General Terms

Algorithms, Experimentation

Keywords

recommender systems, social search, collaborative filtering, proximity measures

1. INTRODUCTION

Recommender systems typically model user preferences based on their past rating behavior and make individualized recommendations from this data. They assume that people who rate a large set of movies similarly will rate all movies similarly. High prediction accuracy in state-of-the-art recommender systems validates this assumption [16]. Relying purely on ratings between users and items strikes us as an overly constraining principle. In fact, recommendation approaches are categorized into content-based [19] and collaborative filtering [20]. Collaborative filtering approaches – those described above – are more accurate than content-based approaches. Hybrid recommender systems take advantage of both of these approaches and typically enjoy even better accuracy [1]. A significant problem with content-based recommendation is that many items may not have any easily indexable content associated with them.

However, a growing amount of data is available on information objects in the form of tags. For example, Delicious[1] is a collaborative tagging database for web pages, Flickr[2] is a tagging site for photos, and IMDb[3] (The Internet Movie DataBase) includes a set of tags that describes movies. Indeed, recent research shows that tagging systems are often descriptive [14], a result that supports using tags as the content features in a recommender system. In the context of such systems, it's interesting to ask whether we can leverage the rich tagging information to make the recommendations more effective than conventional recommender systems. In other words, we believe tags will help recommendation and our main goal is to evaluate a recommender system that includes tags on each item.

In this paper, we propose a graph-based model to achieve this goal. More specifically, we assume there is a matrix

[1]http://del.icio.us
[2]http://www.flickr.com
[3]http://www.imdb.com

of user ratings on items (i.e., a user-item rating matrix) and that each item has a set of tags associated with it. In the interest of generality we do not assume any information about which users tagged which items. Indeed, IMDb is a system where the associations between users, tags and items is not available and our approach can work with such systems. Obviously, whenever such association information were available, then one could take into account all of the joint information about the tags used by particular users. We regard our approach as the minimal reliable data we can assume to be available with only a little effort. That is, we always have ratings data available in collaborate filtering settings. If the system is further a collaborative tagging system, tagging data is available. If tags are not explicitly available in a system, other online resources could be used to generate tags via an information extraction approach. Here, we are not concerned with extracting tags, and we just assume they are available.

In our proposed model, we employ a graph proximity-based approach over a graph consisting of users, items and tags. The edges between users and items can be set according to ratings, and tags are connected to related items. We now have two ways of connecting items, via users and via tags. Thus, unrated items nearby – in a graph proximity sense – the items in a user's rating history should also be interesting to that user. Based on this model, we can sort items according to their graph proximity to the querying user and a group of tags associated with that user. We use an efficient algorithm to approximate the two graph based proximity measures: the Katz score and personalized PageRank. These algorithms are iterative procedures that will converge to the true solution if given enough time, but we terminate them before that point. The approximations usually produce one or two decimal places of accuracy, which is sufficient to get a good sense of the ordering. In other investigations, these approximate approaches have been at least 10 times faster [8].

We begin by reviewing related work in social search and graph based recommendation in the next section. We then describe a simple model to incorporate tags into a recommender system in Section 3. Next, Section 4 describes the proposed model in detail, including the related user graph in Section 4.4. We show how the proposed method works with Netflix Ratings and IMDb tags in Section 5. Throughout much of this research, we were motivated by a vision of a query based recommender system. Section 6 outlines the vision for such a system and discusses future work.

We make the following contributions:

- We propose and implement two approaches to integrate tags and ratings in a recommender system: the first is a straightforward combination of content scores (from tags) and predicted user score (from ratings), see Section 3; the second is novel and employs two commonly used graph proximity measures enhanced by a nearest-neighbor heuristic (Section 4).
- We create a dataset containing both ratings and tags by the joining Netflix and IMDb datasets. Besides its use for our research in validating the utility of graph based proximity measures for enhancing the effectiveness of recommender systems, this "joined" data set is a valuable resource which can be used in future research projects, including extensions of this work (Section 5.1).
- We evaluate the effectiveness of our methods in the context of top-k recommendations. In this context, rather than mean-squared prediction error, measures such as normalized discounted cumulative gain are more appropriate and we use them in our evaluation. The results show our approach is robust, efficient, and effective (Section 5.3).
- We outline a vision for future recommender systems using queries based on tags (Section 6).

2. RELATED WORK

We identified three broad classes of related work: social search, graph based recommendation, and query based hybrid recommendation. A related topic that we do not discuss is personalized search. In our framework, we assume that users provide ratings on the items, which is not the case in most personalized search settings.

2.1 Social Search

The term social search is used to describe a type of web search that considers social information gathered from Web 2.0 applications [5]. Many methods have been proposed to improve discovery of relationships from social data and enhance social search results. For example, [11] proposed FolkRank, a generalized link analysis approach (similar to PageRank), to compute strengths of each entity of the network. FolkRank computes the score of each entity based on its relationships with others and the strengths of the relationships that spread activation. Therefore, a tag stated to be strong by important users becomes strong, and a document strongly related to strong tags by strong users becomes strong itself.

In [2] the popularity of web pages, users, and annotations are captured simultaneously by SocialPageRank based on relationships of entities. The intuition behind this model is that the annotations are good summaries of web pages and popular web pages have higher annotation counts. The contribution of SocialSimRank [2] and SocialPageRank is that combining social score of a page with textual similarity of tags associating that web page to a query improves the quality of the results.

Chakrabarti [6] presents HubRank; a method for proximity searches in entity-relation (ER) graphs, which is fast and space efficient compared to previous proximity search algorithms. HubRank has a preprocessing phase which chooses a small fraction of nodes using query log statistics, and then computes and indexes certain random-walk fingerprints for that fraction of nodes in the multi-entity graph. At query time, a small active subgraph is identified and bordered by nodes with existing indexed fingerprints. These fingerprints are adaptively loaded and the remaining active nodes are then computed iteratively in order to calculate approximate personalized PageRank vectors.

Schenkel et al. [24] expand the scope of social search by collecting social information from LibraryThing[4]. They model social and semantic relationships among tags and items, and calculate the score of a document for a tag for each user based on these relations. The score of a document for a query, then, is produced by summing up the scores of that

[4]http://www.librarything.com

document for tags in the query. Similarly, [23] develop an incremental top-k algorithm considering strengths of relations among users and relations of different tags. They use a top-k threshold model and use social and semantic expansions in an incrementally on-demand manner to leverage social wisdom.

In [5], Carmel et al. try to take searcher's personal preferences into account by re-ranking search results. In order to re-rank search results for a user, they extract related users to that user and compute the similarity strength between them based on their social activity and re-rank the non-personalized search results. Therefore, documents that are strongly related to similar users get boosted in the personalized result.

Zhou et al. [27] combined language-modeling-based methods for information retrieval with social annotations in a unified framework to detect topical information in tags and integrate those information into traditional information retrieval techniques. In the first step they categorize users by domain and extract topics from contents and annotation of documents, and in the second step they incorporate user domain interests and topical background models to enhance document and query language models.

2.2 Graph-Based Recommendation Methods

The relationships between users and items based on their rating preferences can be modeled as a bipartite graph. For example, in a movie recommender system, the nodes of the graph are users and movies, where a user is connected to a movie with a weighted edge if the user rated that movie and the rating is the weight of the edge. Gori et al. ([10]) present ItemRank, a random-walk based scoring algorithm that by using a similarity measure, ranks movies according to expected user preferences. Average commute time, PCA commute time distance, and elements of the graph Laplacian's pseudo-inverse are some of the measures characterizing similarity that Gori et al. used in ItemRank. The intuition behind ItemRank is that user preferences can spread through the correlation graph, so they used the PageRank algorithm because it has both propagation and attenuation properties.

In [9], the authors present a similar method for measuring similarity between any pair of nodes based on the number and length of the paths between them. They compute similarities based on a Markov chain model of random walk through the graph by assigning transition probabilities to the edges and considering items as states of the Markov chain. They show that the pseudo-inverse of the Laplacian matrix of the graph is a valid kernel and can be considered as a similarity measure. Moreover, [4] present various measures and show that commute time is highly sensitive to nodes' degrees, which can be scaled to the stationary distribution of a simple random walk. They propose angular-based measures for recommendation and showed that its performance is much better than using commute time alone. An alternate approach proposed in [12] is to use link prediction measures, including Katz's score, to improve standard the accuracy of collaborative filtering.

Zhang et al. [26] model the label distribution for users and items and also pairwise relationships between users and items as a Gaussian Markov random field. They use this Gaussian semi-supervised model in order to solve the problem of top-k recommendations. Another contribution of their work is using an absorbing random-walk algorithm while considering degrees of nodes and directly generating top-k items without predicting ratings, just like our proposal.

2.3 Query Based Hybrid Recommenders

Many recent commercial movie recommendation systems[5] are designed around the idea of a "movie genome project" – a set of features describing the movies. These were most likely inspired by the success of Pandora's[6] "music genome project" used to build user customized radio-stations. The movie "genes" identified by these systems could easily serve as the tags in our approach. In particular, Jinni implements a query-based search and recommendation system similar to the vision we outline in Section 6.

To improve query based movie search results, [18] combines predicted user ratings with common search methods. In a more general setting, Cheng et al. [7] introduce a model for recommender system where attributes are added to item nodes as new nodes. Items are then sorted based on random walk proximity to a query, which could be a set of item nodes or attribute nodes. Multi-way clustering is used to reduce the amount of computation and hence the effectiveness and efficiency are improved.

3. A SIMPLE MODEL INTEGRATING TAGS

There has been much work on keyword searching within the information retrieval literature. In the most popular scenario, items are ranked via a combination of query-dependent features and document-importance features. The idea is that a less precise match in a highly important document may trump a great match in a total stinker. Common query-based features are the TF-IDF score or the BM25 score [21] between a document and a query. Document-importance features take many forms. Possibly the most well-known are the PageRank scores associated with pages on the web, but domain specific heuristics, such as the number of document views, are equally valid.

Our proposed model for combining tags and recommender systems takes a similar approach to an information retrieval search. Instead of a query, we assume there is a set of tags associated with each user. In our experiments, these are the set of tags on all items the user has rated; but we outline a more expansive vision of true keyword queries for recommender systems in Section 6. Our problem setting is still different from classic keyword search in two main ways. First, our input data is different. In our case we are dealing with two matrices M_U (user/item rating matrix) and M_W (item/keyword occurrence matrix). Second, the ranking must be done with respect to the individual user issuing the query. In this setting, it has more in common with personalized search than standard keyword search, but as pointed out in Section 2, our problem is considerably different from personalized search in taking user ratings into account.

Similar to the ranking described above, we use two main components in our score formula, one of which represents

[5]See http://www.jinni.com, http://www.hellomovies.com, http://www.clerkdogs.com, and http://www.nanocrowd.com. Note that David F. Gleich has a small financial stake in the company behind HelloMovies.

[6]http://www.pandora.com

the score of every item regardless of the tag set but *with respect to the user issuing the query*. Second, we use the TF-IDF score of every item with respect to each tag. Let S_Q represent the TF-IDF score and S_C represent the predicted content score from a collaborative filtering approach. Both score values are then scaled to the $[0, 1]$ interval and linearly combined through a β parameter. Let T represent the set of tags for the user, we define the score associated with user u and item i to be

$$\text{score}(u, i; T) = \beta \cdot S_Q(T, i) + (1 - \beta) \cdot S_C(u, i). \quad (1)$$

This model, however, does not take into account indirect tag similarities. Therefore, it is especially sensitive to tag sparsity. This is due to the fact that every item's score with respect to the tags is calculated only based on those tags that appear in item's set. This might cause problems in situations where the set of tags assigned to every item is not expressive enough to describe all aspects of the item.

4. A GRAPH-BASED MODEL INTEGRATING TAGS

4.1 The Data Structure

The data available to our proposed system is, as mentioned earlier, the user-item rating matrix and a list of tags for each item. We now model the input as a tri-partite graph of user, item, and tag nodes. Edges connect users and items based on the available ratings, while edges between items and tags are simply defined using the item-tag matrix – the M_W matrix. The intuition behind this model is that similarity between items, induced by users or tags, is reinforced.

Let $U = \{u_1, u_2, \ldots, u_m\}$ be the set of users, $I = \{i_1, i_2, \ldots, i_n\}$ be the set of items, and $W = \{w_1, w_2, \ldots, w_k\}$ be the set of tags. Therefore, we can define an undirected graph $G = (V, E)$, where $V = U \cup I \cup W$ is the set of nodes and the adjacency matrix is defined as:

$$A_{i,j} = \begin{cases} 1 & i \in U, j \in I, r_{ij} \geq \mu \\ 1 & i \in I, j \in U, r_{ji} \geq \mu \\ 1 & i \in I, j \in W, t_{ij} = 1 \\ 1 & i \in W, j \in I, t_{ji} = 1 \\ 0 & \text{otherwise} \end{cases}$$

where r_{ij} is the rating of item i_j by user u_i, t_{ij} is the element in i^{th} row and j^{th} column of M_W matrix. This means we add an edge between a user and an item only if the rating of the user on that particular item is at least μ, which in our case is set to 3 (on a scale of 1-5). This setting corresponds to picking all movies that the user "liked", "really-liked", or "loved" according to the Netflix rating scale. Defining edges between items and tags is done in the obvious way using the M_W matrix.

4.2 The Proximity Measures

We are not the first to suggest graph based proximity measures for a recommender system. Bao et al. [2] propose a matrix-algebraic method to propagate similarity of users, items, and tags iteratively until convergence. Sarkar and Moore suggest that an approximated version of commute time could be used to show similarity of users and items in a recommender system [22]. In our proposed graph-based model, we too assume that the proximity of nodes indicates similarity. From the variety of proximity measures in social networks, we have selected the pair-wise Katz measure and the personalized PageRank [17] measure.

The Katz measure between vertices u and v is defined as

$$K(u, v) = \sum_{\ell=1}^{\infty} \alpha^{\ell} \text{Paths}_l(u, v),$$

where $\text{Paths}_l(u, v)$ is the number of paths of length l between two vertices and $0 < \alpha < 1$ is an attenuation factor. Likewise, PageRank is a random-walk-based authority measure defined for nodes in a network. The PageRank score between two nodes is

$$R(u, v) = (1 - \alpha) \sum_{\ell=0}^{\infty} \alpha^{\ell} \text{Prob}_l(u, v)$$

where $\text{Prob}_l(u, v)$ is the probability of random walk starting at u and ending at v in exactly l steps. If we fix v and look at the vector $R(\cdot, v)$, it is the vector of personalized PageRank scores where the reset step always moves to node v.

Recall our setting from the simple model, we assume each user is associated with a set of tags T based on their movies. Therefore, the returned items should be as close to those tags as possible, as well as being close to the user. In order to define a score for each item, suppose that set of user tags is $u_t = \{t_1, t_2, \ldots, t_l\}$, we may define:

$$\text{score}(u, i; T) = \beta \cdot S(u, i) + (1 - \beta) \cdot \sum_{t \in u_t} S(i, t), \quad (2)$$

where $S(j, k)$ is either of the two graph similarity measures defined above.

In order to compute pairwise Katz scores between the user and different items, we use the following approach. The pairwise Katz score between a user (u_i) and all other nodes including all items is a vector (x) found by $x = (I - \alpha \cdot A)^{-1} \cdot e_i$, where e_i is a standard basis vector vector whose i^{th} element is set to 1 and all other elements to 0, and A is the adjacency matrix of the graph. After finding x, we only use those similarity values that correspond to item nodes and normalize the vector. Calculating the similarity between every tag node and every item could be done in the same way.

Unfortunately, computing Katz score using the previous formula with matrix inversion is too expensive for large matrices. Instead, we approximate the solution of the linear system. That is, we define $B = (I - \alpha \cdot A)$, and the Katz score of all nodes with respect to u_i can be computed by solving the following linear system

$$B \cdot x = e_i.$$

However, even this is too expensive. In the next section we describe a technique to approximate the solution of this system by adapting methods for personalized PageRank [3, 15]. This technique only explores a small set of nodes nearby vertex u_i to approximate the largest results.

Personalized PageRank scores satisfy a similar linear system. The PageRank scores between a user (u_i) and all other nodes satisfy

$$(I - \alpha P) \cdot x = (1 - \alpha) e_i,$$

where the matrix P has entries $P_{ji} = 1/d_i$ if node i links to node j, where d_i is the degree of node i. The algorithms in

[3, 15] efficiently estimate the largest elements in x by only exploring a small set of nodes nearby u_i.

4.3 The Algorithm

To approximate the vector of Katz scores, $B \cdot x = e_i$, the algorithm we use is described in [8]. We briefly summarize its features here. Standard iterative methods for large linear systems employ a sequence of matrix-vector products to approximate a solution x. The algorithm from [8] employs a sequence of column queries from B instead. It is inspired by fast techniques for personalized PageRank computation [15, 3] where column queries correspond to out-link queries for a node of the graph. When applied to approximating Katz scores, column queries also correspond to out-link queries. Results from personalized PageRank computation and Katz score computation demonstrate that when solutions of the linear system are localized (meaning only a few entries of x have non-trivial values), these algorithms only access a small set of distinct columns, although they may repeatedly query any particular column. In practice, the computation time for an approximate solution may be only slightly larger than *a single matrix-vector product* [8]. Furthermore, this algorithm does not involve any preprocessing of the graph. It can be implemented straightforwardly whenever there is a technique to access the out-links from a node.

Because the paper proposing this algorithm is still under review, we present the details of the algorithm in Appendix A.

4.4 Nearest Neighbors Heuristic

Since we are only interested in the items that have not been rated by the querying user, those items are connected to the user through paths of at least 3 edges. Therefore, the contribution of these paths to Katz or PPR scores could be insignificant and perhaps lost in numerical computations or approximation. A heuristic method to raise the score of items of potential interest to the user is adding edges from the querying user to a few other user nodes. A common measure for finding these nodes in neighbor-based recommender systems approaches is Pearson correlation coefficient [25]. We add these edges before calculating the proximity measure, and remove them afterwards for queries in the future. The number of non-zeros added to the matrix is negligible, but it can significantly improve the quality. We only find nearest neighbors of the user among those users that have rated at least one common item with the querying user. Moreover, the number of nearest neighbors should be empirically determined, so we took 10 nearest neighbors based on smaller experiments.

5. EXPERIMENTS

5.1 Datasets

In order to run experiments to validate our models, we selected Netflix [16] and IMDb [13] datasets. Netflix provides a collection of user-movie ratings, but it lacks keywords or tags for each movie. Therefore, we joined movies from Netflix to IMDb's database to find a set of keywords for each movie. Our final data contains almost 1 million ratings from 175000 users on 5000 movies, and there are 20000 tags linked to these movies.

We implement two different methods; first the simple method using Pearson correlation as a similarity measure and use a weighted average of ratings of 10 most similar users who have rated the item for predictions. We have also implemented the proposed model exactly as described in Section 4 using Katz and PPR scores, both with and without the nearest neighbors heuristic. We use $\alpha = 10^{-4}$ as the attenuation factor of Katz, $\alpha = 10^{-1}$ for PPR. We vary β from 0 to 1 in increments of 0.1.

5.2 Missing Link Test

We randomly select 1000 users, and for each remove the edge between the user and a random item that has been rated 5. We then apply our approaches to the dataset to retrieve an ordered list of items. For each user, we form the set of tags of all movies they liked. Then we randomly permute this set. We perform tests using the first 1, 4, 7, and 10 tags in the randomly permuted set. By design then, the test with 7 tags includes all the tags used in the test with 4.

Ideally, our algorithm will place the removed item into the top set of results. Thus, we look at the top 10 and 25 items with the highest scores from our method. (These thresholds were chosen because they are common result set sizes on the web.) We then calculate the ratio of the number of times that the removed item appears in the top-k list to the number of trials and call it the *hit rate*. We also compare the run time of different approaches.

5.3 Hybrid Recommender Test

We randomly separate 90% of the ratings for the training set and the remaining 10% for the testing set. From the testing set, we choose 1000 users who have more than 20 ratings in the testing set. For each user, we make a query using all tags of all rated items by the user. We thus have an ordered list of items rated by the users in the test set, according to the actual ratings ($L1$), and retrieve an ordered list of same items according to the scores ($L2$).

We take $L1$ as the ground truth and measure the effectiveness of approaches by comparing $L2$ to it. An effective hybrid recommender system should be able to return an $L2$ list as similar to $L1$ as possible. Among different measures to compare these two list, we have used precision@k, mean average positions (MAP), mean reciprocal ranks (MRR), and normalized discounted cumulative gain (nDCG). Brief description of these measures are in the following:

Precision@k This is simply the ratio of relevant items in the retrieved list to all retrieved items. In calculation of precision@k, we assume items rated 3 and more are relevant.

MAP While precision is a well-known metric for evaluating an ordered list of items returned by a query, MAP is the average of precision@k values for all possible k values.

MRR For each user in the test set, the rank of first related item is desired to be lower. The average of reciprocal of these ranks is thus a proper metric for this study. Items rated 3 or more are considered relevant.

nDCG Discounted cumulative gain is defined as:

$$DCG = rel_1 + \sum_{i=2}^{n} \frac{rel_i}{log_2 i},$$

where $rel_i \in 0, 1$ indicates if i^{th} item is relevant. Normalized DCG is calculated by dividing DCG by the maximum DCG score an algorithm could achieve given the relevancy information beforehand.

5.4 Results

Figure 1 shows that the best β value for combining CF and IR parts of the score (when query size is 1, at which the plot is more demonstrative) is neither 0 nor 1. Therefore, their combination is showing a better performance than the parts alone.

Figures 2 and 3 report top-k hit rate at $k = 10$ and 25 for a few query sizes. As is seen in these figures, the algorithm very well captures the removed item when the query size is around 4 words, but for larger queries, the returned results may be too broad to include the removed item. In scarce queries, PPR, enhanced PPR, and especially the simple model do better than Katz approaches.

Figure 4 shows the run time of different approaches in seconds. Although PPR has a rather better performance, it takes more time to reach the same accuracy as Katz does. It is also obvious that enhancing the algorithm with the nearest neighbors heuristic does not cause a significantly longer run time. Please note that shorter run times of the simple approach is a result of preprocessing the item similarities, which unlike graph-based models, is not very flexible with dynamically changing data.

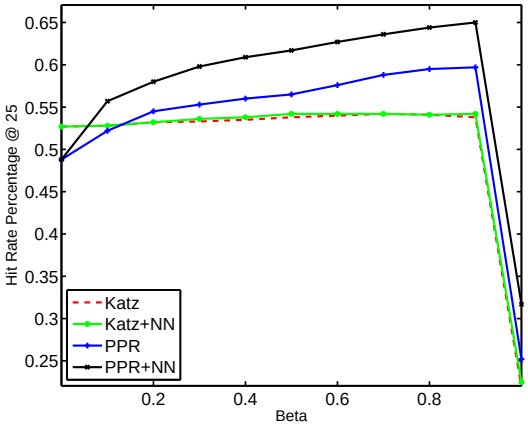

Figure 1: Hit rate vs. β

Results from the hybrid recommender system experiment are shown in Table 1. The nearest neighbors heuristic has slightly improved both Katz and PPR approaches in all the measures used in the experiment. The difference between Katz and PPR is significantly large, and thus Katz is showing a much better quality of returned items compared to the simple model and PPR.

6. OUR VISION: QUERY BASED RECOMMENDATION

Improving the usability of recommender systems and user satisfaction has always been one of the main concerns of the recommender systems community. The popularity of search engines suggests that, in general, users prefer easy-to-use ways of communicating with the system to express

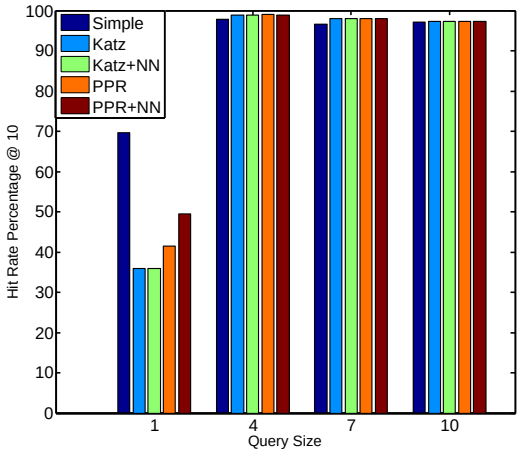

Figure 2: Hit rate of approaches at top-10

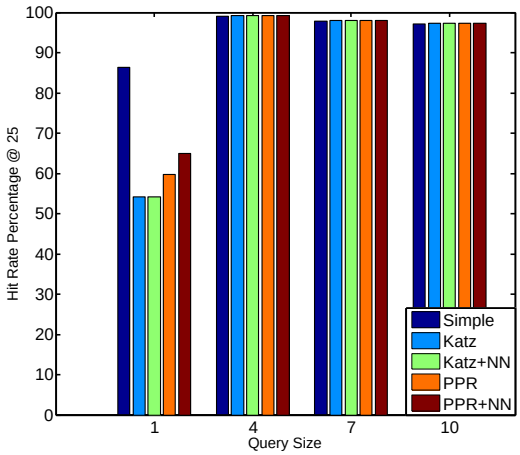

Figure 3: Hit rate of approaches at top-25

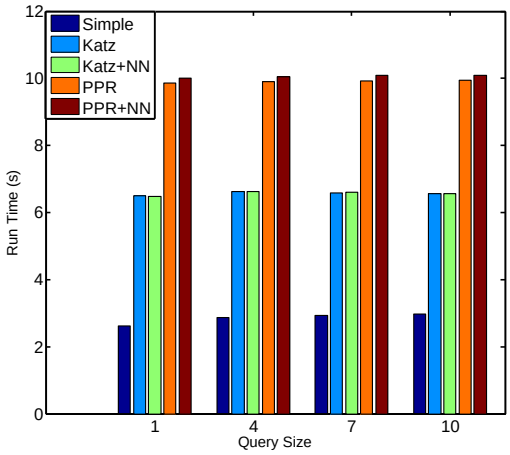

Figure 4: Run time of approaches

	MAP	MRR	Precision	nDCG
Simple	0.0463	0.1606	0.2077	0.0829
Katz	0.1382	0.5874	0.3306	0.2406
PPR	0.0333	0.1002	0.1693	0.0358
Katz+NN	0.1400	0.5912	0.3330	0.2435
PPR+NN	0.0357	0.1051	0.1713	0.0385

Table 1: Comparison of mean average precision (MAP), mean reciprocal rank (MRR), precision@k (k=25), and normalized discounted cumulative gain (nDCG) for different approaches.

their current preferences. Our original motivation was to improve the usability of these system by allowing users to issue keyword queries to a recommender system. For example, a user trying to find a family friendly movie would issue the query "family-friendly." In our system, we could use the tag "family-friendly" as the tag portion of the recommendation instead of the user's implied set of tags (as discussed in the previous two sections). Without this explicit information, we can only infer the current interests of users by complex statistical models, which are not necessarily accurate due to possible lack of sufficient information.

7. CONCLUSION AND FUTURE WORK

In this paper we proposed the idea of combining tags and collaborative filtering in order to improve usability of recommender systems. We first described a simple model and then proposed a graph-based model. We empirically evaluated the approaches in terms of their capacities of performing as hybrid recommender systems using a combination of two real-world datasets and two common proximity measures. We identified the weaknesses and strengths of our approach and provided concrete ideas in order to improve them for the future based on our observations. Finally, we described a vision for recommender systems with keyword queries.

This area is ripe for future work. The current paper is a stepping stone on the path towards the vision for a query enabled recommender system. However, we could refine the current ideas further. Currently, we use all of the tags in the user profile (via the liked movies) as tags used for recommendation. There are certainly better ways of choosing a subset of the most important tags for every user in order to design a better system. One possible way of doing this would be grouping items into two different categories for the user, like and dislike. We could as well group them into multiple categories according to rating levels. Having done this, each of the categories could be considered as a class and the ***mutual information*** between every tag and every class could be used in order to select the most important tags of the user for every class. Using tags that have a high mutual information with "like" class should improve the performance.

Similarly, we can define the keyword query in such a way that results in diversification of recommendations. In order to do this, first we can either cluster the tags or alternatively discover some topics in tags using their co-occurrences in items. Then tag selection could be done again based on the mutual information between the tag and different topics or clusters. Defining a keyword query that contains the best representatives from each topic has the effect of giving items of different types (according to their contents) the opportunity to get the chance of being recommended to the user. On the other hand, the collaborative filtering component of the scoring function will only assign high scores to the items that the user will like and altogether a diverse set of items which are also of user's interest will be returned.

A final limitation of the current system is that there is no way to incorporate information on the movies that a user dislikes. Such information is easy to include in our formulation by subtracting graph proximity scores associated with disliked movies.

We hope to investigate all of these ideas in the future.

8. 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 Trans. on Knowl. and Data Eng.*, 17(6):734–749, 2005.

[2] S. Bao, G. Xue, X. Wu, Y. Yu, B. Fei, and Z. Su. Optimizing web search using social annotations. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 501–510, New York, NY, USA, 2007. ACM.

[3] P. Berkhin. Bookmark-coloring algorithm for personalized PageRank computing. *Internet Mathematics*, 3(1):41–62, 2007.

[4] M. Brand. A random walks perspective on maximizing satisfaction and profit. In *Proceedings of the Fifth SIAM International Conferenceon Data Mining (SDM2005)*, pages 12–19, 2005.

[5] D. Carmel, N. Zwerdling, I. Guy, S. Ofek-Koifman, N. Har'el, I. Ronen, E. Uziel, S. Yogev, and S. Chernov. Personalized social search based on the user's social network. In *CIKM '09: Proceeding of the 18th ACM conference on Information and knowledge management*, pages 1227–1236, New York, NY, USA, 2009. ACM.

[6] S. Chakrabarti. Dynamic personalized PageRank in entity-relation graphs. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 571–580, New York, NY, USA, 2007. ACM.

[7] H. Cheng, P.-N. Tan, J. Sticklen, and W. F. Punch. Recommendation via query centered random walk on k-partite graph. In *ICDM '07: Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, pages 457–462, Washington, DC, USA, 2007. IEEE Computer Society.

[8] P. Esfandiar, F. Bonchi, D. Gleich, C. Greif, L. V. Lakshmanan, and B.-W. On. Fast katz and commuters: Efficient approximation of social relatedness over large networks. Submitted.

[9] F. Fouss, A. Pirotte, J.-M. Renders, and M. Saerens. Random-walk computation of similarities between nodes of a graph with applicaiton to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):355–369, March 2007.

[10] M. Gori and A. Pucci. ItemRank: a random-walk based scoring algorithm for recommender engines. In *IJCAI'07: Proceedings of the 20th international joint conference on Artifical intelligence*, pages 2766–2771, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[11] A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. Information retrieval in folksonomies: Search and ranking. In Y. Sure and J. Domingue, editors, *Proceedings of the 3rd European Semantic Web Conference*, volume 4011 of *Lecture Notes in Computer Science*, pages 411–426, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[12] Z. Huang, X. Li, and H. Chen. Link prediction approach to collaborative filtering. In *JCDL '05: Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries*, pages 141–142, New York, NY, USA, 2005. ACM.

[13] The Internet Movie Database. `http://www.imdb.com`.

[14] C. Körner, D. Benz, A. Hotho, M. Strohmaier, and G. Stumme. Stop thinking, start tagging: tag semantics emerge from collaborative verbosity. In *WWW '10: Proceedings of the 19th international conference on World wide web*, pages 521–530, New York, NY, USA, 2010. ACM.

[15] F. McSherry. A uniform approach to accelerated PageRank computation. In *Proceedings of the 14th international conference on the World Wide Web*, pages 575–582, New York, NY, USA, 2005. ACM Press.

[16] The Netflix Challenge. `http://www.netflixprize.com`.

[17] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford University, November 1999.

[18] S.-T. Park and D. M. Pennock. Applying collaborative filtering techniques to movie search for better ranking and browsing. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 550–559, New York, NY, USA, 2007. ACM.

[19] M. Pazzani and D. Billsus. Learning and revising user profiles: The identification ofinteresting web sites. *Mach. Learn.*, 27(3):313–331, June 1997.

[20] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. GroupLens: an open architecture for collaborative filtering of netnews. In *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, New York, NY, USA, 1994. ACM.

[21] S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In D. K. Harman, editor, *Proceedings of the Third Text REtrieval Conference*, TREC, NIST Special Publication 500-226, pages 109–126, Gaithersburg, MD, November 1994. National Institute of Standards and Technology.

[22] P. Sarkar and A. W. Moore. A tractable approach to finding closest truncated-commute-time neighbors in large graphs. In *Proceedings of the 23rd conference on Uncertainty in Artificial Intelligence (UAI2007)*, 2007.

[23] R. Schenkel, T. Crecelius, M. Kacimi, S. Michel, T. Neumann, J. X. Parreira, and G. Weikum. Efficient top-k querying over social-tagging networks. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 523–530, New York, NY, USA, 2008. ACM.

[24] R. Schenkel, T. Crecelius, M. Kacimi, T. Neumann, J. Xavier Parreira, M. Spaniol, and G. Weikum. Social wisdom for search and recommendation. *IEEE Data Engineering Bulletin*, 31(2):40–49, June 2008.

[25] S. M. Stigler. Francis Galton's account of the invention of correlation. *Statistical Science*, 4(2):73–79, 1989.

[26] Y. Zhang, J. qin Wu, and Y. ting Zhuang. Random walk models for top-n recommendation task. *Journal of Zhejiang University - Science A*, 10(7):927–936, July 2009.

[27] D. Zhou, J. Bian, S. Zheng, H. Zha, and C. L. Giles. Exploring social annotations for information retrieval. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 715–724, New York, NY, USA, 2008. ACM.

APPENDIX

A. TOP-K KATZ SCORES

This section summarizes an efficient algorithm to compute the largest Katz scores with respect to a target node from [8]. Let us begin by stating two conflicting pieces of notation. We use e_i as the vector with a 1 in the ith component and 0 elsewhere. Also, for a vector v, the notation v_i is the scalar in the ith component of v.

Our efficient algorithm for computing the top-k entries of a linear system of the form $B \cdot x = e_i$ is based on the Richardson iteration. To solve $B \cdot x = e_i$ using the Richardson method, we iterate

$$x^{(k+1)} = x^{(k)} + (e_i - Bx^{(k)}).$$

As an aside, we note that the Richardson iteration corresponds with a gradient descent procedure on the problem of minimizing $(1/2)x^T \cdot A \cdot x - x^T \cdot e_i$ over x. If we call $r^{(k)} = e_i - B \cdot x^{(k)}$, which is nothing more than the kth residual or negative gradient, then the iteration is $x^{(k+1)} = x^{(k)} + r^{(k)}$. Rather than taking the entire step along the gradient, a set of PageRank algorithms [15, 3] propose

$$x_i^{(k+1)} = \begin{cases} x_i^{(k)} & \text{if } i \neq j \\ x_i^{(k)} + r_i^{(k)} & \text{if } i = j \end{cases}$$

where j is picked on a criteria discussed below. With this update, $r^{(k+1)} = r^{(k)} - r_j^{(k)} B_j$ and B_j is the jth column of B. Thus, we can solve this linear system by only accessing individual columns of the matrix. As long as the spectral radius of $B < 1$, then this algorithm converges when j is chosen to be the largest magnitude element in $r^{(k)}$. Consequently, we keep the elements of r organized in a heap to permit fast queries of the largest elements.

For the Katz problem, we have $B = I - \alpha A$ where A is a symmetric adjacency matrix of a graph. This matrix produces an especially nice update to the residual. We set $r^{(k+1)} = r^{(k)} - r_j^{(k)} e_j + \alpha r_j^{(k)} B_j$ or

$$r_i^{(k+1)} = \begin{cases} r^{(k)} + \alpha r_j^{(k)} & \text{if } i \text{ is connected to } j \\ 0 & \text{if } i = j \\ r^{(k)} & \text{otherwise.} \end{cases}$$

Thus, the work involved here is proportional to the degree of vertex j. A similar result holds in the PageRank case [15, 3].

We empircally observe that this procedure is fast and efficient at producing approximations of the top-k nodes nearest a target node for the Katz measure [8].