Applying Collaborative Filtering Techniques to Movie Search for Better Ranking and Browsing

Seung-Taek Park
Yahoo! Research
3333 Empire Ave
Burbank, CA 91504
parkst @ yahoo-inc.com

David M. Pennock
Yahoo! Research
45 West 18th St, 6th floor
New York, NY 10011
pennockd @ yahoo-inc.com

ABSTRACT

We propose a new ranking method, which combines recommender systems with information search tools for better search and browsing. Our method uses a collaborative filtering algorithm to generate personal item authorities for each user and combines them with item proximities for better ranking. To demonstrate our approach, we build a prototype movie search and browsing engine called *MAD6* (**M**ovies, **A**ctors and **D**irectors; **6** degrees of separation). We conduct offline and online tests of our ranking algorithm. For offline testing, we use Yahoo! Search queries that resulted in a click on a Yahoo! Movies or Internet Movie Database (IMDB) movie URL. Our online test involved 44 Yahoo! employees providing subjective assessments of results quality. In both tests, our ranking methods show significantly better recall and quality than IMDB search and Yahoo! Movies current search.

Categories and Subject Descriptors

H.3.5 [**Information Systems**]: On-line Information Services; H.3.4 [**Information Systems**]: Systems and Software; H.3.3 [**Information Systems**]: Information Search and Retrieval

General Terms

Experimentation, Measurement, Performance

Keywords

collaborative filtering, search ranking, recommender systems, information retrieval, movie search

1. INTRODUCTION

Two types of technologies are widely used to overcome information overload: *information retrieval* and *recommender systems*. Information retrieval systems (e.g., general web search engines such as Google[1] and Yahoo! Search[2]) accept a query from a user and return the user relevant items against the query. Since the number of returned documents can run into the millions, a good ranking algorithm, which ensures high precision in the top ranked documents, is important for the success of an information retrieval system.

In general, the ranking of returned documents in web search engines is the combination of the document proximity and authority. Document proximity, sometimes called document relevance, denotes the document's similarity or relevance to the given query. Document authority denotes the importance of a document in the given document set. PageRank [19] measures global importance of documents on the Web while HITS [16] measures local authorities and hubs in the base-set documents extracted by the given query. However, even though item authority and proximity are widely used together in general search engines for better document ranking, item authority is often ignored or partially used in many specialized search systems, for example product search in an ecommerce site. For example, search results may be sorted based only on item relevance against the given query.

There are several challenges for adapting item authority in these information retrieval systems due to the different characteristics of documents like item or product information documents in commercial sites, as compared to web documents. The power of PageRank and HITS stems from the feature of links between web documents. PageRank and HITS assume that a link from document i to j represents a recommendation or endorsement of document j by the owner of document i. However, in item information pages in commercial sites, links often represent different kinds of relationships other than recommendation. For example, two items may be linked because both items are produced by the same company. Also, since these item information pages are generally created by providers rather than users or customers, the documents may contain the providers' perspective on the items rather than those of users or customers.

On the other hand, recommender systems are widely used in ecommerce sites to overcome information overload. Note that information retrieval systems work somewhat passively while recommender systems look for the need of a user more actively. Information retrieval systems list relevant items at higher ranks only if a user asks for it (e.g. when a user submits a query). However, recommender systems predict the need of a user based on his historical activities and recom-

KDD'07, August 12–15, 2007, San Jose, California, USA.
Copyright 2007 ACM 978-1-59593-609-7/07/0008 ...$5.00.

[1] http://www.google.com

[2] http://search.yahoo.com

mend items that he may like to consume even though the user does not specifically request it.

In this study, we propose a new approach to combine informational retrieval and recommender system for better search and browsing. More specifically, we propose to use collaborative filtering algorithms to calculate personalized item authorities in search. This approach has several benefits. First, user ratings or behavior information (e.g., user click logs) better represent user's recommendation than links in the item information pages. Second, this information is biased to the customers' perspectives on items rather than those of providers. Third, many ecommerce sites provide users both information retrieval and recommender systems. Calculating item authorities using these already existing recommender systems in ecommerce sites does not require much additional work and resources. Fourth, using both item authorities and proximities, search results can be improved. Last, since collaborative filtering algorithms provide personalized item authorities, the system can provide a better personalized user experience.

To demonstrate our approach, we build a prototype personalized movie search engine called MAD6. The name is an acronym for **M**ovies, **A**ctors, and **D**irectors with **6** degrees of separation.[3] MAD6 combines both information retrieval and collaborative filtering techniques for better search and navigation. MAD6 is different from general web search engines since it exploits users' ratings on items rather than the link structures for generating item authorities. Moreover, using the users' historical preference data and expected preferences on items, MAD6 provides a personalized search ranking for each user.

To evaluate our ranking algorithms, we conduct offline and online tests. We use search click logs from Yahoo! Search for the offline test. We extract queries that resulted in a user clicking on a movie page in either the Internet Movie Database (IMDB) or Yahoo! Movies. We compare the performance of five search ranking systems, including IMDB search, Yahoo! Movies current search, and three of our own algorithms, according to hit rate and reciprocal ranking. We also conduct an online test involving 44 Yahoo! employees using our test demo. The users subjectively rate the recall and quality of six ranking systems. Our approach performs better than IMDB search and Yahoo! Movies current search in both the offline and online tests, according to both recall and quality of search results.

2. RELATED WORK

Recommender systems can be built in three ways: content-based filtering, collaborative filtering, and hybrid systems. Content-based recommender systems, sometimes called information filtering systems, use behavioral user data for a single user in order to try to infer the types of item attributes that the user is interested in. Collaborative filtering compares one user's behavior against a database of other users' behaviors in order to identify items that like-minded users are interested in. Even though content-based recommender systems are efficient in filtering out unwanted information and generating recommendations for a user from massive information, they find few if any coincidental discoveries. On the other hand, collaborative filtering systems enables serendipitous discoveries by using historical user data.

Collaborative filtering algorithms range from simple nearest-neighbor methods [5, 25, 27] to more complex machine learning based methods such as graph based methods [1, 15], linear algebra based methods [4, 26, 10, 24, 7], and probabilistic methods [14, 22, 23, 8]. A few variations of filterbot-based algorithms [11, 21] and hybrid methods [23, 18, 3] that combine content and a collaborative filtering have also been proposed to attack the so-called *cold-start* problem.

Tapestry [9] is one of the earliest recommender systems. In this system, each user records their opinions (annotations) of documents they read, and these annotations are accessed by others' filters. GroupLens[4] [25], Ringo [28] and Video Recommender [30] are the earliest fully automatic recommender systems, which provide recommendations of news, music, and movies. *PHOAKS* (People Helping One Another Know Stuff) [29] crawls web messages and extracts recommendations from them rather than using users' explicit ratings. *GroupLens* has also developed a movie recommender system called *MovieLens*[5] [11, 26, 27]. *Fab* [2] is the first hybrid recommender system, which use a combination of content-based and collaborative filtering techniques for web recommendations. Tango [6] provides online news recommendations and Jester [10] provides recommendations of jokes.

Page et al. [19] and Kleinberg [16] first proposed a new concept of document relevance, often called document authority, and developed the PageRank and HITS algorithms, respectively, for better precision in web search. Both algorithms analyze the link structure of the Web to calculate document authorities. Haveliwala [12] proposed *topic-sensitive PageRank*, which generates multiple document authorities biased to each specific topic for better document ranking.

Note that our approach is different from general web search engines since we use user ratings rather than link structure for generating item authorities. Also, our approach is different from topic-sensitive PageRank since we provide personalized item authorities for each user rather than topic-biased item authorities. Also, our approach is different from recommender systems since it uses predictions of items as a ranking function for information search rather than generating recommendation.

3. RANKING ALGORITHM

Note that we only focus on movie title search rather than people (actor or director) search in our study. Thus, the term "item" is equivalent to "movie" or "movie title". Like general web search engines, our ranking algorithm consists of two main components: item proximity and item authority.

3.1 Item proximity: DB and Web relevance

3.1.1 DB relevance

Most movie search engines index only titles or few keywords on items. Thus, item relevance for the given query against a database are often measured by relevances of titles and keywords for the query. In other words, they are

[3] 6 degrees of separation is a well-known phrase from sociology adapted more recently to the movie domain in the form of a "party game" called "six degrees of Kevin Bacon", where the goal is to identify as short a path as possible from a given actor to Kevin Bacon, following co-actor links.

[4] GroupLens, http://www.grouplens.org/

[5] MovieLens, http://movielens.umn.edu/

most useful when users already know what they are looking for. Search queries are assumed to be part of movie titles, or names of actors or directors. We define these type of queries as navigational queries.

However, when a user searches for something, in many cases he does not know much about the object, and that is one of main reasons why he searches for it. Sometimes, searching means trying to find unknown (or unfamiliar) information, which may be interesting. Thus, search tools should anticipate that some queries will be ambiguous or inexact. Even for niche search engines, the situation is not changed. Imagine a scientific literature search. Even though a scientist is very familiar with her research field, sometimes she is searching for articles that she might have missed. In this case, we cannot expect that she already knows the titles of the articles she is looking for.

A fan of "Arnold Schwarzenegger" may try to find a list of the actor's movies with a query such as "arnold action", expecting to find movies such as *The Terminator* or *Conan the Barbarian*. We define these type of queries as informational queries. However, the Internet Movie Database (IMDB) and Yahoo! Movies, for example, currently do not return these movies, since their titles do not contain any of the query words. Since both systems' basic search supports title and name matching only, they suffer from poor recall when a user does not know the exact titles of the target movies. Another example of a poorly supported query type is for character names. Users may want to find *The Lord of the Rings* series with queries such as "gandalf" or "frodo". IMDB does provide a character name search option in their advanced search, but only one name is allowed and gender information is required. Thus, the query "neo trinity" (looking for *The Matrix*) is not supported. Yahoo! Movies does not support character name search at this time.

To address these limitations, we build our own database of more extensive metadata for better recall in search results. In addition to movie titles, we index a large amount of metadata including genres, names of actors, directors, characters, plots, MPGA ratings, award information, reviews of critics and users, captions from trailers and clips, and so on. To measure item relevance for the given query, we use MySQL's `match-against` function in each indexed field. The function returns matched items with relevance scores in each field and we calculate item relevances for the query by calculating the weighted sum of all fields. A few heuristics are used to balance the weight of each field. For example, we give more weight on the title field, so that items with title matches will have higher relevance scores. Relevance scores of the returned documents are normalized such that the highest score in each search becomes 13. (In our system, relevance scores are integers from 1 to 13, corresponding to ratings grades F through A+.)

To show the possible increase in recall, we conducted a recall test comparing our extensive indexing system with IMDB and Yahoo! Movies current search. We downloaded movie data from IMDB[6] and generated queries for the 100 most popular movies, where popularity is measured by the number of user ratings. We use five movie metadata fields: names of actors, directors, characters, plots, and genres. The two highest TF/TFIDF words (except stopwords) of each of the top 100 popular movies are selected as a query and only the top 10 returned movies are analyzed. We consider only movies which exist in our database.

Table 1: Hit ratios of three movie search engines for the top 100 most popular movies; Only the top 10 returned movies are considered. "DB" denotes our base system with an extensive index. Two top TF/TFIDF terms from five metadata, including names of actors, directors, characters, plots, and genres, are selected as a query for each movie in the top 100 popular movies. Then each query is submitted to three systems, IMDB, Yahoo! Movies and our base system. The popularities of movies are measured by the number of user ratings. We downloaded the IMDB movie content data and conducted this test in April 2006.

	TF		TFIDF	
	HIT	No Returns	HIT	No Returns
IMDB	4	2	6	2
current Yahoo!	2	94	2	95
DB	**33**	25	**37**	43

Only a few queries succeed in extracting the target movie within top 10 highest position when IMDB and current Yahoo! Movies are used. All of the successful queries contain at least one title word. Table 1 shows the improvement of our base search engine with an extensive index. Note that queries are somewhat biased toward IMDB, since they were generated based on IMDB data. Our system successfully returned the target movie about 1/3 of the time, whereas IMDB and current Yahoo! Movies returned the target movie less than 6% of the time. Note that IMDB conducts OR matching search and returns many matches in most cases. Yahoo! Movies conducts AND matching search and our system conducts AND matching search for items but OR matching search for the names.[7]

The performance gain of our system is much smaller when tested with actual user queries on Yahoo! Movies instead of synthetic queries. This is because most queries on Yahoo! Movies current search are navigational: all or part of movie titles or actor names. However, it is impossible to know whether users inherently prefer navigational queries, or if users have learned through experience that informational queries are not supported by today's search systems at IMDB and Yahoo! Movies. We expect the frequency of informational queries to increase significantly when movie search engines start to support such queries. Though we employ synthetic queries here, we report results of performance comparisons with real queries and real users in Sections 4 and 5.

3.1.2 Web relevance

We also use an additional concept of item relevance for the given query that we call *web relevance*, which takes into account the relevance as judged by a general purpose algorithmic search engine like Yahoo! or Google. We find that users often provide some extra information of items which

[6] http://www.imdb.com/interfaces

[7] OR matching search returns any items that match at least one query term while AND matching search returns items that match all query terms.

Table 2: The effect of web relevance. "jlo" is submitted to each system as a query. Bold represents items relevant to Jennifer Lopez. The results are as of April 2006.

System	Top 10 movie results
Yahoo!	No returns
IMDB	1. Fejlvs (1968)
	2. Mihajlo Bata Paskaljevic (2001)
	3. Mihajlo Petrovic Alas (1968)
	4. Mijlocas la deschidere (1979)
	5. Scopul si mijloacele (1983)
	6. A Szr s gykr fejldse (1961)
	7. Ziveo zivot Tola Manojlovic (1973)
	8. Kumonosu j (1957)
	9. The Hi-Lo Country (1998)
	10. Vlemma tou Odyssea, To (1995)
DB	No returns
Web	1. **Maid in Manhattan (2002)** A+
	2. **Angel Eyes (2001)** A-
	3. Let's Dance (1950) B+
	4. Sweet 15 (1996) B+
	5. My Family (1995) B+
	6. **U-Turn (1997)** B+
	7. **The Cell (2000)** B
	8. **The Wedding Planner (2001)** C-

do not exist in our database. For example, "jlo"—the nickname of actress Jennifer Lopez—is often found in the users' reviews of the movies she has starred. Moreover, the performance of general search engines are constantly improving, reflecting a huge number of person-hours of development. In fact, performing a Yahoo! or Google search restricted to a particular movie site (e.g., querying a search engine with "arnold action site:movies.yahoo.com") often works better than using the site-specific search on the movie site itself. One of the advantages for this approach is that we can take advantage of any improvements made by general search engines without delay, without re-inventing all the tools and tricks that they use.

We use the Yahoo! Search API[8] for getting web information. Each time our system gets a query from a user, it conducts a site-limited search through the API and gets the top 50 results. Then, our system grabs the item ids from the document URLs and extracts information of corresponding items from our database. The web relevance score of each returned item is given based on its relative first position[9] in the web search result. More specifically, if $L(i, q)$ is the highest rank of an item i in the web search result for the query q, its web relevance score is given by the following equation.

$$Web(i, q) = \frac{(N + 1 - L(i, q))}{N} * \gamma \tag{1}$$

where N and γ are the maximum number of returns from the search engine and a normalized factor. We set $\gamma = 13$ such that the web relevance score of the top ranked item becomes 13. We set $N = 50$. We ignore all items that do not appear in our database. Table 2 shows the effect of Web relevance for the query "jlo". Our web relevance system returns eight items and five of them are relevant to Jennifer Lopez. IMDB does not return any relevant titles in its top 10 search results.

3.1.3 Combining of DB and Web relevance

The item proximity score of a returned document is calculated as:

$$Prox(i, q) = max(Web(i, q), DB(i, q)) \tag{2}$$

where $DB(i, q)$ and Web(i,q) denote DB and Web relevances of an item i for the given query q. We tested several heuristic weighting schemes such as averaging two relevance scores or selecting worst relevance score as the proximity scores and found that this heuristic method seems to be the best among them. Note that the definition of a good ranking is very subjective and we do not have clear metric for this matter. Thus we always depend on the judgment of the first author when parameters are chosen. In other words, our implemented ranking system is very biased to the taste of the first user. We do not claim that our ranking is the best performing instance among all possible implementations. Our goal of this study is showing that collaborative filtering can be useful for ranking when PageRank can not be used for the item authority.

3.2 Item authority

3.2.1 Global item authorities

We first generate global item authorities. The global item authorities can be generated based on the items' average ratings over all users. However, we add some heuristics for calculating global item authorities, which emphasize both the popularity and quality of items. Note that the quality of items do not always match with the need of users. For example, even though some old movies have very good quality, most users may not look for those 40's or 50's movies since they prefer recently produced movies. In fact, only 57 users in our database have rated *Citizen Kane (1941)*. Thus, we calculate global item authorities using the following equation:

$$Auth_i = \frac{\overline{r_i} + log_\gamma|U_i| + \overline{c_i} + log_{10}(10 * aw_i + 5 * an_i)}{\delta} \tag{3}$$

where U_i is the set of users who have rated item i, $\overline{r_i}$ is the average rating of item i over all users, $\overline{c_i}$ is the average critic rating of item i, aw_i is the number of awards that item i has won, an_i is the number of awards that item i has been nominated for, and δ is a normalization factor such that the maximum global item authority is 13. Also, we set γ such that the maximum value of $log_\gamma|U_i|$ is 13. We use award scores and average critic ratings on items for assigning better authorities to the classic movies than the movies of which users have frequently rated but their average ratings are low.

3.2.2 Personal item authorities: Prediction

We use an item-based collaborative filtering (CF) algorithm [27] to calculate a user's expected ratings on the returned items. We have tested several collaborative filtering algorithms including user-based CF and a few machine

[8] http://developer.yahoo.com/search/web/

[9] Note that the returned items can be duplicated. One may be a "Cast and Credit" page and another may be a "Movie Details", "Showtimes & Tickets", or "Trailers & Clips" page of the same movie.

Table 3: Weak & strong generalization: The average normalized mean absolute errors and standard deviations on three sample sets are shown. Smaller numbers (lower errors) are better.

Data	Generalization	Item-Based	MMMF
MovieLens	Weak	.4096 ±.0029	.4156 ±.0037
	Strong	.4113 ±.0104	.4203 ±.0138
EachMovie	Weak	.4382 ±.0009	.4397 ±.0006
	Strong	.4365 ±.0024	.4341 ±.0025

learning algorithms and found that item-based was the best performing collaborative filtering algorithm among them over three movie data sets including Yahoo! Movies, MovieLens, and EachMovie. Table 3 shows the normalized mean absolute error of item-based CF and maximum margin matrix factorization (MMMF), the best algorithm among ten tested machine learning approaches in [17, 24], according to the weak and strong generalization tests with the MovieLens and the EachMovie data. The results of MMMF are copied from [24]. More motivations and details of the weak and strong validation test can be found in [17, 24].

Item-based CF first calculates item similarities using *adjusted cosine similarity*:

$$sim(i,j) = \frac{\sum_{u \in U}(r_{u,i} - \overline{r}_u) \cdot (r_{u,j} - \overline{r}_u)}{\sqrt{\sum_{u \in U}(r_{u,i} - \overline{r}_u)^2} \cdot \sqrt{\sum_{j \in U}(r_{u,j} - \overline{r}_u)^2}} \qquad (4)$$

where $r_{u,i}$ is the rating of user u for item i and $\overline{r}_u$ is user u's average item rating. It helps to penalize similarity scores that are based on the small number of common users in order to reflect less confidence, yielding a modified similarity score sim'(i,j) as follows [13]:

$$sim'(i,j) = \frac{min(|U_i \cap U_j|, \gamma)}{\gamma} * sim(i,j) \qquad (5)$$

where U_i denotes a set of users who have rated the item i. We set $\gamma = 50$. We use user rating information from Yahoo! Movies[10] to calculate item similarities. The prediction of the target item for the user is given by the sum of the average rating of the target item and the weighted average of its neighbors:

$$p_{u,i} = \overline{r}_i + \frac{\sum_{j \in I_u} sim'(i,j) * (r_{u,j} - \overline{r}_j)}{\sum_{j \in I_u} |sim'(i,j)|} \qquad (6)$$

where $\overline{r_i}$ and I_u denote the average rating of the item i over all users and a set of items the user u has rated.

3.3 MADRank: Our ranking system

We assign item authorities for each search result based on the following procedure. We assign global item authorities as item authorities when the target user is unknown. When a user logs in our system, we partition returned items in each search result into two groups: items which the user has rated and others that the user has not rated. We assign the user's own ratings as item authorities for the first group and the user's expected ratings calculated by item-based algorithm for the second group. If we cannot calculate the user's expected ratings for any items in the second group due to lack of information, global item authorities are assigned for those items. Then the ranking score of document i for the given query q and user u is:

$$MADRank(i,q,u) = \alpha * Auth(i,q,u) + (1-\alpha) * Prox(i,q) \qquad (7)$$

where α is an weighting factor for item authorities. We set $\alpha = 0.5$. In addition, we set the MADRank score to 13 if the title of an item exactly matches to the given query.

Table 4 shows the top 10 title search results of six movie search systems, including the current Yahoo! Movies search, IMDB search, and four of our own search systems, for the query "arnold action". **DB** denotes one variant of our systems with an extensive index and DB relevance based ranking. **Web** denotes a system using the Yahoo! Search API and web relevance ranking. **GRank** denotes a system using MADRank as a ranking system and item authorities are based on global item authorities. **PRank** denotes a system with MADRank and personal item authorities. Table 5 shows the profile of the test user used in the PRank.

Note that the current Yahoo! Movies search does not return any titles due to AND matching within a limited index. IMDB does not return any Arnold Schwarzenneger movies in the top ten results. In the DB system, *Arnold Schwarzenneger DVD 2-Pack - The Sixth Day/The Last Action Hero(2003)* is shown first since the title contains both "arnold" and "action". However, the results still show the need of better ranking for informational search since many famous titles such as *Terminator (1984)* and *Terminator 2: Judgment Day (1991)* do not appear in the first search results. The result of Web seems to be better than that of DB. Note that several famous Schwarzenneger movies including *Terminator (1984)* and *Terminator 3 (2003)* appear in the top 10 results. However, *Arnold Schwarzenneger DVD 2-Pack (2003)* is still shown first. Several items including *Terminator (1984)* and *Terminator 2 (1991)* are boosted in the GRank due to their higher item authorities while *Arnold Schwarzenneger DVD 2-Pack (2003)* disappears from the top 10 due to its low global item authority. In PRank, *Terminator (1984)*, *Terminator 2 (1991)* and *Total Recall (1990)* are boosted further since either the user has rated the items higher or his expected ratings for those items are high. Similarly, *Terminator 3 (2003)*, *Eraser (1996)* and *End of Days (1999)* disappear due to their low personalized item authorities. By applying item authorities in the ranking function, we believe that search results can be significantly improved.

4. OFFLINE EVALUATION

We measure the effectiveness of our ranking algorithms by comparing against two existing systems—IMDB search and the current Yahoo! Movies search—in both online and offline tests. In this section, we discuss our offline test. We use search click data from Yahoo! Search containing search queries and clicked URLs. We select data stored on the first day of each month from October 2005 to May 2006. We extract all queries that resulted in a click on a movie page in either IMDB or Yahoo! Movies. Specifically, we extract queries containing clicked-URLs starting with `imdb.com/title` or `http://movies.yahoo.com/shop?d=hv`. We use some heuristics to extract additional movie pages

[10] User ratings of movies consist of a small sample generated by Yahoo! Movies on November 2003. The data contains 211,327 ratings, 7,642 users and 11,915 items. All users rate at least 10 movies.

Table 4: Top 10 results of different ranking methods for the query "arnold action". The results are as of April 2006.

Ranking	Top 10 movie results
current Yahoo!	No items return
IMDB	1. Einleitung zu Arnold Schoenbergs .. (1973) 2. Benedict Arnold: A Question of Honor (2003) 3. Love and Action in Chicago (1999) (V) 4. Mary-Kate and Ashley in Action! (2001) 5. Armed for Action (1992) 6. Demonstrating the Action of the ... (1900) 7. Peace Is Every Step: Meditation ... (1998) 8. Rock 'n' Roll Space Patrol Action Is Go! (2005) 9. Leibstandarte SS-Adolf Hitler im .. (1941) 10. Action Figures: Real and Uncut (2005) (V)
DB	1. Arnold Schwarzenneger DVD 2-Pack - The Sixth Day/The Last Action Hero(2003) 2. THE LAST ACTION HERO (1993) and the 2 - DVD Special Edition of THE 6TH DAY 3. Warner Home Video DVD Action 4-Pack (1997) 4. Last Action Hero (1993) 5. The 6th Day (2000) 6. Eraser (1996) 7. Commando (1985) 8. True Lies (1994) 9. Nancy Drew - A Haunting We Will Go (1977) 10. Out for Justice (1991)
Web	1. Arnold Schwarzenneger DVD 2-Pack - The Sixth Day/The Last Action Hero(2003) 2. Last Action Hero (1993) 3. Commando (1985) 4. End of Days (1999) 5. Eraser (1996) 6. True Lies (1994) 7. Terminator 2 - Judgment Day (1991) 8. Raw Deal (1986) 9. Terminator 3: Rise of the Machines (2003) 10. Collateral Damage (2002)
GRank	1. True Lies (1994) 2. Last Action Hero (1993) 3. Commando (1985) 4. Terminator 2 - Judgment Day (1991) 5. End of Days (1999) 6. Eraser (1996) 7. The Terminator (1984) 8. The Bridge on the River Kwai (1957) 9. Terminator 3: Rise of the Machines (2003) 10. The Fugitive (1993)
PRank	1. Terminator 2 - Judgment Day (1991) 2. Commando (1985) 3. True Lies (1994) 4. Last Action Hero (1993) 5. The Terminator (1984) 6. T2 The Ultimate Edition DVD (1991) 7. The Bridge on the River Kwai (1957) 8. Bloodsport (1988) 9. Total Recall (1990) 10. The Fugitive (1993)

Table 5: The profile of the test user

Air Force One (1997) (F)
Commando (1985) (C+)
Hulk (2003) (C-)
Lord of the Rings: The Fellowship of the Ring (2001) (A)
Lord of the Rings: The Return of the King (2003) (A)
Matrix (1999) (A+)
Raiders of the Lost Ark (1981) (A)
Return of the Jedi (1983) (B-)
Saving Private Ryan (1998) (A)
Shawshank Redemption (1994) (A+)
Star Wars (1977) (A+)
Terminator (1984) (A)
Terminator 2: Judgment Day (1991) (A+)

and remove cast and crew pages. We extract $110K \sim 170K$ queries for each day.

We choose 500 randomly selected instances from each day for a total of 4,000 query-URL pairs. Next, we extract movie ids from the URLs. If a URL is a Yahoo! Movies page, we extract the yahoo movie id from the URL and find the movie title in our database. If a URL is an IMDB page, we submit the URL to Yahoo! Search and find the title, then find the matching Yahoo! movie id. If this fails, we use some additional heuristics. For example, if we cannot find a corresponding movie by title match (i.e. Bom yeoreum gaeul gyeoul geurigo bom (2003) in IMDB), we submit a query such as "site:movies.yahoo.com Bom yeoreum gaeul gyeoul geurigo bom (2003)" to Yahoo! Search and select the first returned movie as its counterpart in Yahoo! Movies (e.g. Spring, Summer, Fall, Winter... and Spring (2004) in Yahoo! Movies). Then, the first author manually inspected the match list and removed a few incorrect instances. After removing all instances without either Yahoo! or IMDB IDs, we are left with 2,179 query-movieID pairs.

For the purpose of the test, we assume that the clicked movie URL is the user's desired target movie for that query. We test five ranking systems, including IMDB search, Yahoo! Movies current search, and three of the methods described in the Section 3.3. We cannot test the personalized algorithm PRank, because we do not have access to the movie ratings of users in our search click log data set.

Our database includes movie meta data and user rating information from Yahoo! Movies, including 21M user ratings, 3.3M users and 150K movies. This data is also used for the current version of MAD6 implementation and online test discussed in Section 5.

We use *hit rate* (**HR**) and *average reciprocal hit rank* (**ARHR**) [8] as performance metrics for the offline tests. Hit rate measures how many times a system returns the target movie using following equation:

$$HR = \frac{|H|}{|N|} \tag{8}$$

where N is a set of test instances and H is a set of hit instances within the top 10 results.

Note that our test data contains, for each query, a list of potentially several movies clicked by the same user. When a user clicks on several movies from the same query, we count a hit if the system returns any of the visited movies.

Table 6: Offline experiment results.

Metric	IMDB	Yahoo!	DB	Web	Grank
HR	.7435	*.5241*	.5879	.8086	**.8155**
$ARHR$	.6544	.4615	*.4479*	**.7585**	.7303

Then average reciprocal hit rank is measured by the following equation:

$$ARHR = \frac{1}{|N|} \sum_{i \in H} \frac{1}{r_i} \qquad (9)$$

where r_i is the actual rank of the target movie i. When a user visits several movies with the same query, we only consider the highest ranked item. We only consider the top 10 results with this metric. Note that hit rate is used to capture recall of search results while average reciprocal hit rank is used to measure the quality of search results.

The results are shown in Table 6. In general, we find that Web and GRank perform better than the others. GRank performs best on hit rate while Web performs best on ARHR. Even though DB shows a much better hit rate than Yahoo!, DB performs worst on ARHR. Yahoo! performs worst in terms of hit rate due to its AND matching search. IMDB performs better than Yahoo! and DB, but worse than Web and GRank in general.

One may be somewhat surprised by the high hit rate of IMDB and Web. This happens since most queries are navigational queries. In other words, most queries are either movie titles, part of movie titles, or movie titles with extra terms such as the word "movie". We believe this happens because most users understand from experience that informational queries are not supported by existing search facilities, and thus mainly submit only supported (i.e., navigational) queries. Also note that our offline test is biased to Web ranking, since we use Yahoo! Search click data. If a target movie is not found by a search engine, a user refines his/her query and submits it again to the search engine. In our test, the first failure would not be captured, and queries that tend to work well in web search (i.e., that result in clicks on movie pages) are tested. Also our test may be somewhat biased to IMDB, since most URLs in our test data come from IMDB. In our test data, 1,297 unique URLs point to IMDB pages while 231 unique URLs point to pages in Yahoo! Movies.

5. ONLINE EVALUATION

We also conduct an online evaluation using MAD6 [20], our prototype personalized movie search and browsing engine. In this section, we first briefly explain MAD6, then discuss our online test procedure and results.

5.1 MAD6 architecture

The architecture of MAD6 is shown in Figure 1. It has four internal components (User Interface (UI) Module, Database, Web Analyzer and Ranker) and two external components (Search Engine and Collaborative Filtering (CF) Module). Note that the two external components are modular and can be exchanged with other systems. The User Interface (UI) Module gets a query from a user and presents the user the search results from the Ranker. When the UI Module obtains a query from a user, it passes the query to the Web Analyzer and Database. The Web analyzer extracts the web search result from the associated search engine and generates web relevance scores of the returned items. Then, this information is submitted to the Database. The Database extracts items relevant to the given query and generates their DB relevance scores. It also extract all information of items extracted by Web Analyzer or Database itself. The information contains item contents, global item authorities and DB and Web relevances. Then, this information is submitted to the Ranker, which requests the CF Module expected ratings of items for the given user. Note that CF Module has its own user rating database. Then, items are sorted based on the ranking scheme the user has requested.

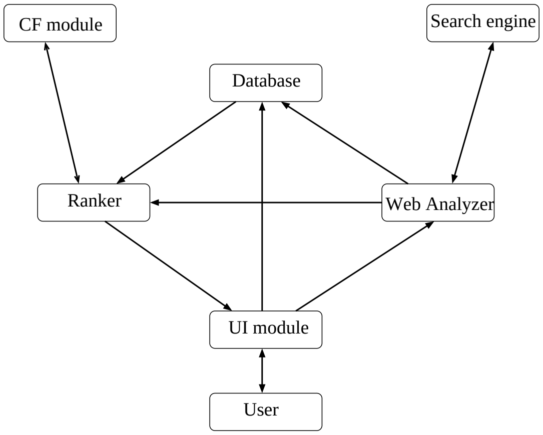


Figure 1: The architecture of MAD6.

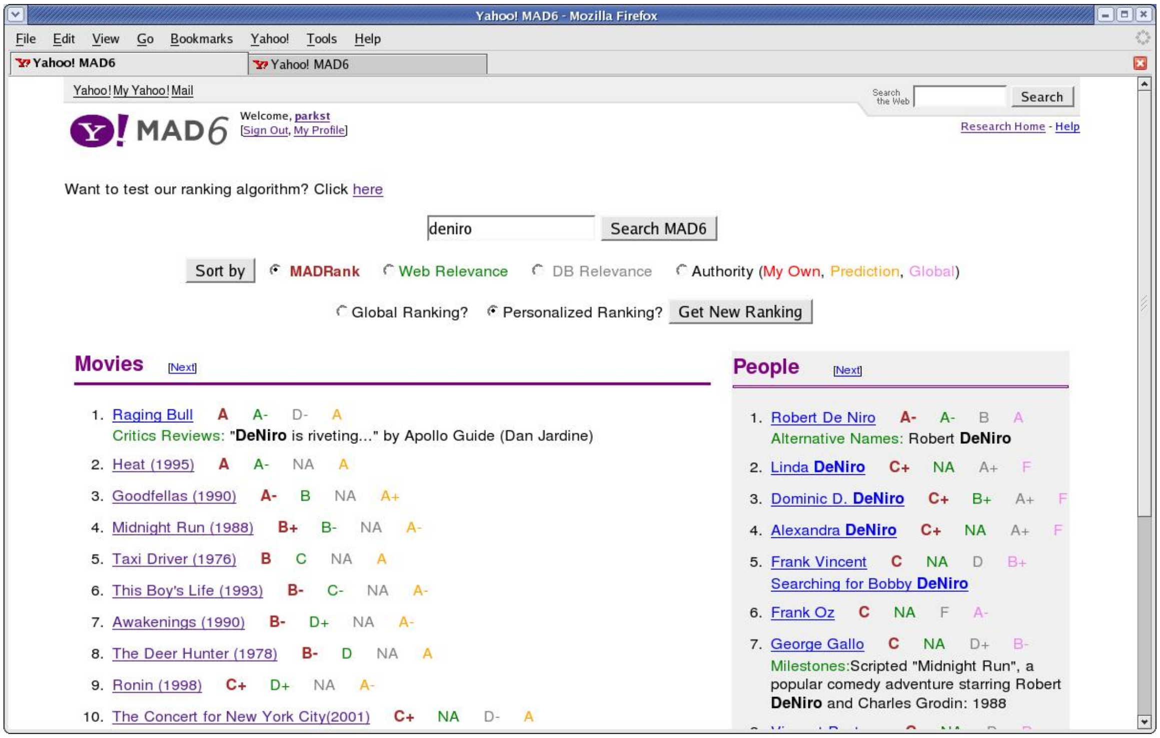
5.2 Features of MAD6

MAD6 provides users three information features; **Search**, **Item Presentation**, and **Personal User Profile** pages. In *Search* pages, MAD6 presents two search results for a given query: movie search results and people search results. Search ranking can be personalized if a user logs in the system. The user can choose to rank results according to global MADRank, personalized MADRank, Web relevance, DB relevance, or item authorities. Each returned item shows ratings based of four methods (MADRank, Web relevance, DB relevance and item authorities) and matched fields against the given query. An example search result is shown in Figure 2.

There are two types of *Item Presentation* pages: movies pages and people (actor and director) pages. Each page shows details of the item (title, synopsis, cast, release date, ratings, posters, etc.) along with two lists of relevant items, one showing neighboring items in the collaboration graph of actors and directors, the other based on similarities inferred from user preferences.

The *Personal User Profile* page presents the user personal information such as: (1) What queries has the user submitted most frequently? (2) What movies, actors and directors has the user visited most frequently, either directly or indirectly?[11] (3) What are the user's favorite genres? (4) What movies are most recommended for the user? Park et al. [20] describe and illustrate MAD6 in greater detail.

[11] By an *indirect visit* we mean visiting a movie or person that links to the movie or person in question via the movie graph.



Welcome, parkst
[Sign Out, My Profile]

Want to test our ranking algorithm? Click here

deniro

● MADRank ○ Web Relevance ○ DB Relevance ○ Authority (My Own, Prediction, Global)

○ Global Ranking? ● Personalized Ranking?

Movies

1. Raging Bull A A- D- A
   Critics Reviews: "**DeNiro** is riveting..." by Apollo Guide (Dan Jardine)
2. Heat (1995) A A- NA A
3. Goodfellas (1990) A- B NA A+
4. Midnight Run (1988) B+ B- NA A-
5. Taxi Driver (1976) B C NA A
6. This Boy's Life (1993) B- C- NA A-
7. Awakenings (1990) B- D+ NA A-
8. The Deer Hunter (1978) B- D NA A
9. Ronin (1998) C+ D+ NA A-
10. The Concert for New York City(2001) C+ NA D- A

People

1. Robert De Niro A- A- B A
   Alternative Names: Robert **DeNiro**
2. Linda **DeNiro** C+ NA A+ F
3. Dominic D. **DeNiro** C+ B+ A+ F
4. Alexandra **DeNiro** C+ NA A+ F
5. Frank Vincent C NA D B+
   Searching for Bobby **DeNiro**
6. Frank Oz C NA F A-
7. George Gallo C NA D+ B-
   Milestones:Scripted "Midnight Run", a popular comedy adventure starring Robert **DeNiro** and Charles Grodin: 1988

Figure 2: Search Result

5.3 Test procedure

Our online test consists of 44 Yahoo! employees submitting 180 relevance judgments between October 25 2006 and November 7 2006. The test is conducted with our test demo shown in Figure 3.

Each participant is asked to rate at least 10 movies using MAD6 before they participate in the test. When a user comes to our test demo, the user is asked to either submit any free-form query, or to select one of 60 suggested queries provided by our system. The 60 queries are randomly selected from about 5,000 pre-selected queries, including the 500 most popular movie titles, 1,000 actors and 1,000 characters (the top 2 actors/roles from each of the top 500 most popular movies) and 2,400 randomly selected Yahoo! Search queries that resulted in a click on a movie link (300 randomly selected queries from each day in our data set).

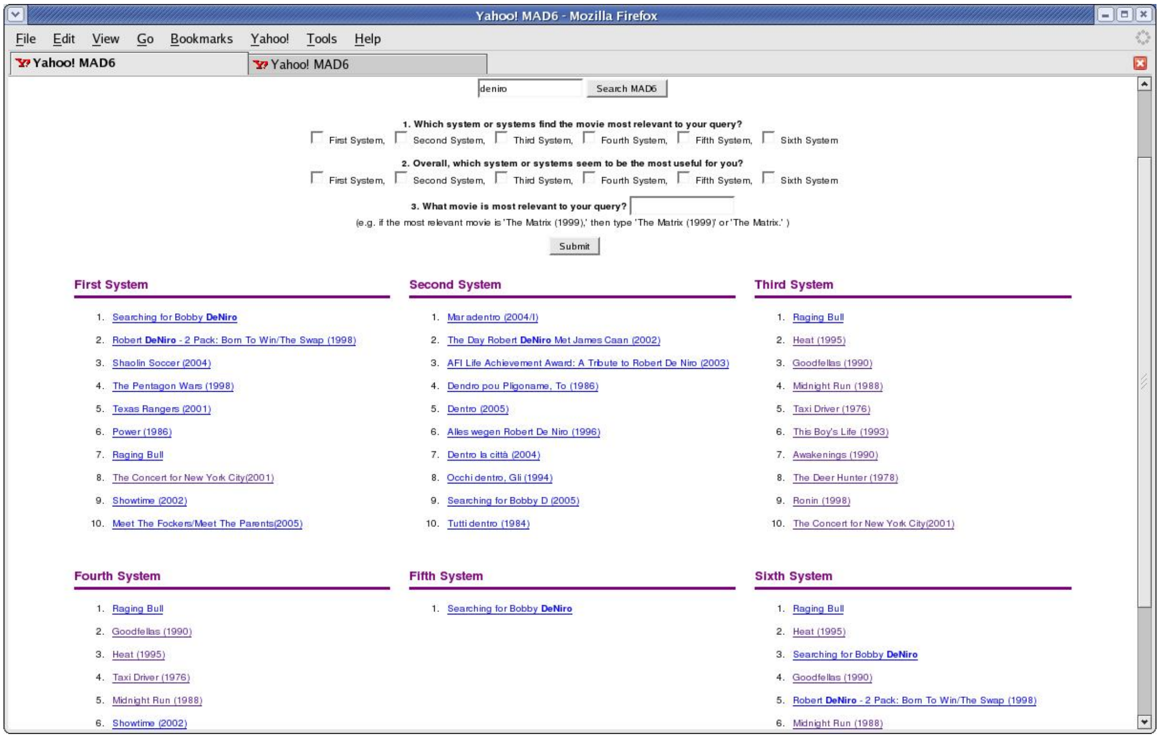
After a participant submits a query, the test demo returns ranked results of six systems: IMDB search, Yahoo! Movies current search, and our four algorithms. The demo does not tell users which result is generated by which system, and the location of each system is randomly selected whenever a query is submitted. After reviewing the six different search results, participants are asked to answer following three questions: (1) Which system or systems finds the movie most relevant to your query? (2) Overall, which system or systems seems to be the most useful for you? and (3) What movie is most relevant to your query? A participant may select multiple systems for the first two questions. The first question is asked to measure recall of search results and the second question is asked to measure quality of search results. The third question requires a free form text answer.

5.4 Test results

In general, we find that Web, GRank and PRank perform better than IMDB, current Yahoo! Movies, and DB, as shown in Table 7. PRank performs best on recall while GRank performs best on quality of search results when all 180 relevance judgments are analyzed. We also classify queries by comparing them with the user's stated target movie (question 3). If a query exactly matches with the title of the target movie, we consider it a navigational query. Otherwise, we consider it an informational query. This classification is manually done by the first author. We classify 49 queries as navigational queries.

A feature of IMDB search hampers our ability to evaluate IMBD results for navigational queries. For exact title matches, IMDB often (but not always) returns the inferred target movie page directly, rather than listing one or more movies on a search results page. Our wrapper did not properly handle this case and the online test showed empty results for IMDB when this happened. Thus, we exclude IMDB when results of navigational queries are evaluated. Note that we did make the proper correction for the offline test in the previous section.

When navigational queries are submitted, PRank performs best while DB performs worst both on recall and quality of search results. It is somewhat surprising that the recall of most systems remains about 50 ∼ 60% even when the titles of the target movie is submitted as a query. We find that many participants often check only one system in the first question even though all six systems return the target movie, probably because participants either did not inspect results carefully enough or misinterpreted the intent of question 1. Thus, we want to point out that our recall analysis on the online test may contain some noise. When


Yahoo! MAD6 - Mozilla Firefox
File Edit View Go Bookmarks Yahoo! Tools Help
Yahoo! MAD6
Yahoo! MAD6
Search MAD6
Submit

deniro

1. Which system or systems find the movie most relevant to your query?

□ First System, □ Second System, □ Third System, □ Fourth System, □ Fifth System, □ Sixth System

2. Overall, which system or systems seem to be the most useful for you?

□ First System, □ Second System, □ Third System, □ Fourth System, □ Fifth System, □ Sixth System

3. What movie is most relevant to your query?

(e.g. if the most relevant movie is 'The Matrix (1999),' then type 'The Matrix (1999)' or 'The Matrix.')

First System

1. Searching for Bobby **DeNiro**
2. Robert **DeNiro** - 2 Pack: Born To Win/The Swap (1998)
3. Shaolin Soccer (2004)
4. The Pentagon Wars (1998)
5. Texas Rangers (2001)
6. Power (1986)
7. Raging Bull
8. The Concert for New York City(2001)
9. Showtime (2002)
10. Meet The Fockers/Meet The Parents(2005)

Second System

1. Mar adentro (2004/I)
2. The Day Robert **DeNiro** Met James Caan (2002)
3. AFI Life Achievement Award: A Tribute to Robert De Niro (2003)
4. Dendro pou Pligoname, To (1986)
5. Dentro (2005)
6. Alles wegen Robert De Niro (1996)
7. Dentro la città (2004)
8. Occhi dentro, Gli (1994)
9. Searching for Bobby D (2005)
10. Tutti dentro (1984)

Third System

1. Raging Bull
2. Heat (1995)
3. Goodfellas (1990)
4. Midnight Run (1988)
5. Taxi Driver (1976)
6. This Boy's Life (1993)
7. Awakenings (1990)
8. The Deer Hunter (1978)
9. Ronin (1998)
10. The Concert for New York City(2001)

Fourth System

1. Raging Bull
2. Goodfellas (1990)
3. Heat (1995)
4. Taxi Driver (1976)
5. Midnight Run (1988)
6. Showtime (2002)

Fifth System

1. Searching for Bobby **DeNiro**

Sixth System

1. Raging Bull
2. Heat (1995)
3. Searching for Bobby **DeNiro**
4. Goodfellas (1990)
5. Robert **DeNiro** - 2 Pack: Born To Win/The Swap (1998)
6. Midnight Run (1988)

Figure 3: Test demo

informational queries are submitted, PRank shows the best recall while GRank performs best on quality.

It is interesting that even though there is not much difference between informational and navigational queries on recall, we find a big difference on quality of search results. When navigational queries are submitted, participants are more satisfied with PRank and Web than GRank. However, when informational queries are submitted, participants prefer GRank rather than PRank and Web. One possible explanation is that, when participants submit navigational queries, they may have very clear target movies in their minds. These movies may be their favorites and are more likely rated before the test. In this case, the authority of the target movies are very accurate since they are real ratings provided by the participants. However, when informational queries are submitted, participants may not have clear target movies and returned items may have less probabilities to be rated. Also, users may search for somewhat unfamiliar items. For example, a user may search for some good romantic comedy movies for dating, even though he has mostly watched and rated action movies. Then item authorities calculated by the item-based algorithm may be inaccurate due to the lack of user information.

Note that only 6 participants rated 20 or more movies and most participants rated fewer than 15 movies. So our online test environment may be considered a *cold-start user* setting where all test users are relatively new users. This cold-start environment may reduce the quality of PRank's personalized search results, since the item-based algorithm suffers from a cold-start problem. We believe that users' satisfaction of PRank will increase as users provide more ratings. To support this argument, we also measure recall and quality of search results based on different participant groups. We find that participants who have rated at least 20 movies are more satisfied with PRank while participants with less than 20 ratings prefer GRank. It is also interesting that the quality of Web results are almost always similar to that of PRank.

6. FUTURE WORK

We plan to incorporate some of the search and browsing features of MAD6 in Yahoo! Movies and related Yahoo! properties. We plan to develop a pseudo natural language query interface ("shortcuts on steroids") for supporting simple question and answering. For example, we would like to be able to handle queries like: "Who won the best actor Oscar in 1995?", or "highly rated comedy starring Arnold Schwarzenegger". Moreover we would like to answer some personalized questions such as "Recommend me an action movie from 2005" or "Who is my favorite 90s actress?". We plan to use MAD6 as a online research platform for testing various search, browsing, personalization, and recommendation algorithms and interfaces. We also plan to utilize our hybrid recommendation algorithm [21] to provide better cold-start predictions and recommendations.

7. CONCLUSIONS

In this paper, we discuss our new ranking method, which combines recommender systems and search tools for better informational search and browsing. To evaluate our approach, we have built MAD6, a personalized movie search engine with some unique features. In both offline and online tests, MAD6 seems to provide users better search recall and quality than IMDB search and Yahoo! Movies current search by combining proximities and authorities of the returned

Table 7: Online experiment results. The "Recall" column lists the percentages of users who selected the corresponding system in answering question 1. The "Quality" column lists the percentages for question 2.

Test group	Systems	Recall	Quality
All (180 feedbacks from 44 users)	IMDB	*22.2*	15.6
	Yahoo!	22.8	*13.9*
	DB	36.7	20
	Web	52.2	38.3
	GRank	53.9	**40**
	PRank	**57.8**	38.3
Navigational queries (49 feedbacks from 23 users)	IMDB	42.9	34.7
	Yahoo!	53.1	30.6
	DB	*44.9*	*22.4*
	Web	57.1	34.7
	GRank	59.2	28.6
	PRank	**63.3**	**36.7**
Informational queries (131 feedbacks from 40 users)	IMDB	14.5	8.4
	Yahoo!	*11.5*	*7.6*
	DB	33.6	19.1
	Web	50.4	39.7
	GRank	51.9	**44.3**
	PRank	**55.7**	38.9
Users with at least 20 ratings (28 feedbacks from 6 users)	IMDB	*14.3*	*10.7*
	Yahoo!	*14.3*	*10.7*
	DB	28.6	14.3
	Web	50	**35.7**
	Grank	**57.1**	28.6
	PRank	**57.1**	**35.7**
Users with less than 20 ratings (152 feedbacks from 38 users)	IMDB	*23.7*	16.4
	Yahoo!	24.3	*14.5*
	DB	38.2	21.1
	Web	52.6	38.8
	Grank	53.3	**42.1**
	PRank	**57.9**	38.3

items. Even though MAD6 is one application in the movie domain, we believe that our approach is general enough to apply other domains including music, travel, shopping and web search.

8. ACKNOWLEDGMENTS

We thank Yahoo! Movies for providing movie content and user rating information. We thank Dennis DeCoste, Gunes Erkan, Rosie Jones, Bernard Mangold, and Omid Madani. We thank the Yahoo! employees who participated in our online test.

9. REFERENCES

[1] C. C. Aggarwal, J. L. Wolf, K.-L. Wu, and P. S. Yu. Horting hatches an egg: a new graph-theoretic approach to collaborative filtering. In *ACM KDD*, pages 201–212, 1999.

[2] M. Balabanovic and Y. Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.

[3] J. Basilico and T. Hofmann. Unifying collaborative and content-based filtering. In *ICML*, 2004.

[4] D. Billsus and M. J. Pazzani. Learning collaborative information filters. In *ICML*, pages 46–54, 1998.

[5] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI*, pages 43–52, 1998.

[6] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin. Combining content-based and collaborative filters in an online newspaper. In *ACM SIGIR Workshop on Recommender Systems*, 1999.

[7] D. DeCoste. Collaborative prediction using ensembles of maximum margin matrix f actorization. In *ICML*, 2006.

[8] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM TOIS*, 22(1):143–177, Jan 2004.

[9] D. Goldberg, D. Nichols, B. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[10] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.

[11] N. Good, J. B. Schafer, J. A. Konstan, A. Borchers, B. M. Sarwar, J. L. Herlocker, and J. Riedl. Combining collaborative filtering with personal agents for better recommendations. In *AAAI/IAAI*, pages 439–446, 1999.

[12] T. Haveliwala. Topic-sensitive pagerank. In *WWW*, pages 517–526, 2002.

[13] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *ACM SIGIR*, pages 230–237, 1999.

[14] T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *IJCAI*, pages 688–693, 1999.

[15] Z. Huang, H. Chen, and D. Zeng. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM TOIS*, 22(1):116–142, Jan 2004.

[16] J. Kleinberg. Authoritative sources in a hyperlinked environment. In *ACM-SIAM Symp. Discrete Algorithms*, pages 668–677, 1998.

[17] B. Marlin. *Collaborative filtering: A machine learning perspective.* Master's thesis, University of Toronto, Computer Science Department, 2004.

[18] P. Melville, R. Mooney, and R. Nagarajan. Content-boosted collaborative filtering. In *AAAI*, 2002.

[19] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[20] S.-T. Park, D. M. Pennock, and D. DeCoste. Applying collaborative filtering techniques to movie search for better ranking and browsing. In *ITWP*, 2006.

[21] S.-T. Park, D. M. Pennock, O. Madani, N. Good, and D. DeCoste. Naíve filterbots for robust cold-start recommendations. In *KDD*, 2006.

[22] D. Pennock, E. Horvitz, S. Lawrence, and C. L. Giles. Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach. In *UAI*, pages 473–480, 2000.

[23] A. Popescul, L. Ungar, D. Pennock, and S. Lawrence. Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. In *UAI*, pages 437–444, 2001.

[24] J. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *ICML*, 2005.

[25] P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *ACM CSCW*, pages 175–186, 1994.

[26] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Application of dimensionality reduction in recommender systems–a case study. In *ACM WebKDD Workshop*, 2000.

[27] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW*, pages 285–295, 2001.

[28] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating ”word of mouth”. In *CHI*, 1995.

[29] L. Terveen, W. Hill, B. Amento, D. McDonald, and J. Creter. PHOAKS: A system for sharing recommendations. *Communications of the ACM*, 40(3):59–62, 1997.

[30] M. R. W. Hill, L. Stead and G. Furnas. Recommending and evaluating choices in a virtual community of use. In *ACM CHI*, pages 194–201, 1995.