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

Dennis DeCoste
Yahoo! Research
3333 Empire Ave
Burbank, CA 91504
decosted@yahoo-inc.com

Abstract

In general web search engines, such as Google and Yahoo! Search, document relevance for the given query and item authority are two major components of the ranking system. However, many information search tools in ecommerce sites ignore item authority in their ranking systems. In part, this may stem from the relative difficulty of generating item authorities due to the different characteristics of documents (or items) between ecommerce sites and the web. Links between documents in an ecommerce site often represent relationship rather than recommendation. For example, two documents (items) are connected since both are produced by the same company. 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 engine called *MAD6* (**M**ovies, **A**ctors and **D**irectors; **6** degrees of separation).

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 a search engine.

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 (Page *et al.* 1998) measures global importance of documents on the web while HITS(Kleinberg 1998) 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 in search results, item authority is often ignored or partially used in many search systems in ecommerce sites. For example, search results are often sorted based on only 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 in commercial sites (e.g., item or product information documents) from web documents. The power of PageRank and HITS mainly comes 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 some kind of relationship rather than recommendation. For example, two items may be linked since 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 providers' perspectives 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 the his historical activities and recommend 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

[1]http://www.google.com
[2]http://search.yahoo.com

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 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. Even though we apply our ranking method to one specific domain, we believe that our ranking approach is general enough and it can be applied to other domains, including web search, by using fast and scalable collaborative filtering algorithms.

Related Work

Page et al. (1998) and Kleinberg (1998) first proposed a new concept of document relevance—often called document authority—and proposed PageRank and HITS algorithms for better precision in web search (Kleinberg 1998; Page *et al.* 1998). Both algorithms analyze the link structure of the web and calculate document authorities similarly. Later, Haveliwala (2002; 2003) proposed *topic-sensitive PageRank*, which generates multiple document authorities biased to each specific topic for better document ranking.

Recommender systems can be built in three ways: content-based filtering, collaborative filtering and hybrid recommender 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, it cannot find any coincidental discoveries. For example, a user may like "Star Wars" even though he/she dislikes most "Harrison Ford" movies. If a system filters out all "Harrison Ford" movies based on the user's profile, then the user will not have a chance to find "Star Wars". On the other hand, collaborative filtering systems enables serendipitous discoveries by using historical user data.

Collaborative filtering systems can be divided into two classes: memory-based and model-based algorithms (Breese, Heckerman, & Kadie 1998). Memory-based algorithms (Resnick *et al.* 1994; Breese, Heckerman, & Kadie 1998) store all historical user information in memory and use a statistical technique to find a set of closest neighbors of the target user. Then, the system combines the preferences of neighbors to generate predictions of unrated items. Model-based algorithms first build a model of user ratings. This model can be built by using Bayesian networks (Breese, Heckerman, & Kadie 1998), clustering (Breese, Heckerman, & Kadie 1998; Ungar & Foster 1998), or classifiers (Billsus & Pazzani 1998; Miyahara & Pazzani 2000).

Collaborative filtering algorithms range from the simple nearest-neighbor methods (Breese, Heckerman, & Kadie 1998; Resnick *et al.* 1994; Sarwar *et al.* 2001) to more complex machine learning based methods such as graph based methods (Aggarwal *et al.* 1999; Huang, Chen, & Zeng 2004), linear algebra based methods (Billsus & Pazzani 1998; Sarwar *et al.* 2000; Goldberg *et al.* 2001; Marlin & Zemel 2004; Rennie & Srebro 2005; DeCoste 2006), and probabilistic methods (Hofmann & Puzicha 1999; Pennock *et al.* 2000; Popescul *et al.* 2001; Karypis 2001; Deshpande & Karypis 2004). A few variations of filterbot-based algorithms (Good *et al.* 1999; Park *et al.* 2005) and hybrid methods (Balabanovic & Shoham 1997; Popescul *et al.* 2001; Melville, Mooney, & Nagarajan 2002; Basilico & Hofmann 2004a; 2004b) that combine content and a collaborative filtering have also been proposed to attack the cold start problem.

Tapestry (Goldberg *et al.* 1992) 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] (Resnick *et al.* 1994; Konstan *et al.* 1997; Miller, Riedl, & Konstan 1997), Ringo (Shardanand & Maes 1995) and Video Recommender (W. Hill & Furnas 1995) are the earliest fully automatic recommender systems, which provide recommendations of news, music, and movies. *PHOAKS* (People Helping One Another Know Stuff) (Terveen *et al.* 1997) crawls web messages and extracts recommendations from them rather than using users' explicit ratings. *GroupLens* also have developed a movie recommender system, called *MovieLens*[5] (Good *et al.* 1999; Sarwar *et al.* 2000; 2001; Rashid *et al.* 2002; McNee *et al.* 2003). *Fab* (Balabanovic & Shoham 1997) is the first hybrid recommender system, which use a combination of content-based and collaborative filtering techniques for web recommendations. Tango (Claypool *et al.* 1999) provides online news recommendations and Jester (Goldberg *et al.* 2001) provides recommendations of jokes.

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

[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/

is different from topic-sensitive PageRank since we provide personalized item authorities for each users 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.

Ranking algorithm

Like general web search engines, our ranking algorithm consists of two main components: item proximity and authority.

Item proximity; DB and Web relevance

DB relevance We found that 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 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 his research field, sometimes he or she is searching for articles he or she might have missed. In this case, we cannot expect that he or she already knows the titles of the articles he or 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, 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 coverage 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 using mySQL, which supports an extensive metadata indexing for better recall in search result. In other words, we index not only titles of movies, but other metadata such as genres, names of actors, directors and characters, plots, MPGA ratings, award information, reviews of critics and users, captions, and so on.[6] 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 score. Relevance scores of the returned documents are normalized such that the highest score in each search becomes 13.[7] We also use another heuristic scheme such that the relevance score becomes 13 if the title of an item exactly matches the given query.

Table 1: Hit ratios of three movie search engines for the top 100 most popular movies; Only the top 10 returned movies are considered. The EI system denotes our simple system with extensive indexing. Queries are generated based on TF/TFIDF descending order. The popularities of items are measured by the number of user ratings.

	TF		TFIDF	
	HIT	No Returns	HIT	No Returns
IMDB	4	2	6	2
Yahoo! Movies	2	94	2	95
EI system	**33**	25	**37**	43

To show the possible increase in coverage, we conducted a performance test comparing our extensive Indexing system with IMDB and Yahoo! Movies. We first downloaded movie data from IMDB[8] and generated queries for the top 100 popular movies. Popularity of movies is measured by the number of user ratings. We use three movie metadata: names of actors, directors and characters, plots, and genres of movies. 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.

Only a few queries succeed to extract the target movie within top 10 highest position when IMDB and Yahoo! Movies are used. All of the successful queries contain at least one title word. Table 1 shows the performance improvement of our simple search engine with extensive indexing in terms of coverage. Note that generated queries are somewhat biased toward IMDB, since they were generated based on IMDB data. Our simple system successfully returned the target movie about 1/3 of the time, whereas IMDB and 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

[6] Movie data was obtained from Yahoo! Movies and contains ratings and metadata for movies opening on or before November 2003.

[7] In our system, relevance scores are integers from 1 (F) to 13 (A+).

[8] ftp://ftp.fu-berlin.de/pub/misc/movies/database/

names.

Web relevance We also introduce a new concept of item relevance for the given query, called *web relevance*. We find that users often provide some extra information of items which 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 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[9] for getting web information. Each time our system get a query from a user, it conducts a site-limited search through the API and get the top 50 results. Then, our system grabs the item ids from the document URLs and extract corresponding items from our database. The web relevance score of each returned item is given based on its relative first position in the web search result. More specifically, if an item i first appears in the k_ith position in the web search result for the query q, its web relevance score is given by the following equation.

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

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 $N = 50$ since the API only provides the top 50 results for the query. Then, the item proximity score of a returned document is calculated as:

$$Prox(i,q) = max(Web(i,q), DB(i,q)) \qquad (2)$$

where $DB(i,q)$ denotes DB relevance of an item i for the given query q. We tested several heuristic weighting schemes for getting better item proximity score and found that this heuristic method seems to be the best among them. Table 2 shows the effect of Web relevance for the query "jlo." Our web relevance system returns 9 items and 6 items are relevant to "Jennifer Lopez." IMDB returns 21 items—2 items for companies, 12 items for names and 7 items for titles—but no items are relevant.

Item authority

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 use our heuristics for calculating the global item authorities, which emphasizes both 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 40s or 50s' movies since they prefer recently produced movies. In fact, only 57 users users in our database have rated "Citizen Kane (1941)." Thus, we calculate global item authorities using the following heuristic equation:

$$Auth_g(i) = \frac{\overline{r_i} + log_2(|U_i|) + \overline{c_i} + log_{10}(10 * aw_i + 5 * an_i)}{k} \qquad (3)$$

where $\overline{r_i}$, U_i, $\overline{c_i}$, aw_i, an_i and k denotes the average rating of the item i over all users, a set of users who have rated the item i, the average critic rating of the item i, the number of awards the item i has won, the number of awards the item i has nominated and a normalized factor such that the maximum global item authority becomes 13, respectively. Also, we force the maximum rating of each of three factors to be 13. For example, if $log_{10}(10 * aw_i + 5 * an_i)$ is more than 13, we give 13 for this factor. 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.

Personal item authorities We use an item-based collaborative filtering algorithm (Sarwar *et al.* 2001) to calculate a user's expected ratings on the returned items. Actually, we built several collaborative filtering algorithms including user-based and a few machine learning based algorithms and conducted performance tests using mean absolute error as a performance metric. We found that item-based was the best performing collaborative filtering algorithm among them. In fact, Park et al.(2005) shows that the item-based algorithm is still one of the best pure CF algorithms by comparing its performance with MMMF, a new and well regarded CF algorithm (Rennie & Srebro 2005), according to the weak and strong generation test. The results in the Table 3 are copied from (Park *et al.* 2005).

Note that the item-based algorithm can be considered as a model-based approach consisting of two parts: an item similarity computation (or model learning stage) and neighbor selection (or prediction calculation stage) using the the model. For example, the item-based algorithm 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 (Herlocker *et al.* 1999):

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

where U_i denotes a set of users who have rated the item i. We set $\gamma = 50$. Note that this process can be done offline. We use user rating information from Yahoo! Movies[10] to calculate item similarities.

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

[10] User ratings of movies consist of a small sample generated by

Table 2: The effect of web relevance. Bold represents items relevant to "Jennifer Lopez."

Query	DB relevance	Web relevance	Yahoo! Movies	IMDB
jlo	No returns	1. **Maid in Manhattan (2002)** A+ 2. **Angel Eyes (2001)** A- 3. Let's Dance (1950) B+ 4. **Jennifer Lopez** B+ 5. Sweet 15 (1996) B+ 6. My Family (1995) B+ 7. **U-Turn (1997)** B+ 8. **The Cell (2000)** B 9. **The Wedding Planner (2001)** C-	No returns	companies (2/2) 1. JLOpen Productions 2. Robert Vukajlo Productions names (5/12) 1. Tatyana Samojlova 2. Miki Manojlovic 3. Branko Mihajlovski 4. Mihailo Cagic 5. Yuri Mikhajlov titles (5/7) 1. Fejlvs (1968) 2. Mihajlo Bata Paskaljevic (2001) 3. Mihajlo Petrovic Alas (1968) 4. Mijlocas la deschidere (1979) 5. Scopul si mijloacele (1983)

Table 3: Weak & strong generalization: The average NMAE and the standard variations on three sample sets are shown.

Algorithm	MovieLens		EachMovie	
	Weak	Strong	Weak	Strong
MMMF	.4156 $\pm$.0037	.4203 $\pm$.0138	.4397 $\pm$.0006	.4341 $\pm$.0025
Item-based	.4096 $\pm$.0029	.4113 $\pm$.0104	.4382 $\pm$.0009	.4365 $\pm$.0024

Then when a new user-item pair comes to the system, the algorithm selects the top k nearest neighbors of the target item from the user's historical preference profile—items the user has rated— by using the item similarity matrix. Then 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:

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

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.

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. 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:

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

where α is an weighting factor for item authorities. We set $\alpha = 0.5$.

Table 4 shows the top 10 title and name search results of six movie search systems, including Yahoo! Movies, IMDB and four of our own, for the query "arnold action." "Extensive indexing" denotes one variant of our systems with extensive indexing and DB relevance based ranking. "Web relevance" 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 Yahoo! Movies does not return any titles or names due to the limited indexing. IMDB returns 21 items including 8 companies, 11 names and 2 titles[11], but all returned items are not relevant to "Arnold Schwarzenneger." In the "extensive indexing" 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 result still shows 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 relevance" seems to be better than that of "extensive indexing" system since search engines use some kind of item authority concept for their ranking algorithms. Sev-

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.

[11] We do not show the search results for companies from IMDB in Table 4.

Table 4: Top 10 results of different ranking methods for the query “arnold action”

Ranking	Top 10 movie results	Top 10 name results
Yahoo! Movies	No returns	No returns
IMDB	1. Einleitung zu Arnold Schoenbergs Begleitmusik zu einer Lichtspielscene (1973) aka “Introduction to Arnold Schoenberg’s Accompaniment to a Cinematic Scene” 2. Benedict Arnold: A Question of Honor (2003) (TV)	1. Anton Arnold 2. Arnold Antonin 3. Antonio T. Arnold Jr. 4. Martin Arnold (I) 5. Anna Antonovskaya 6. Martin Arnold (II) 7. Arnold Jackson 8. Marion Arnold 9. Arnold MacDonald 10. Arnold Labaton
Extensive indexing	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)	1. Horacee Arnold 2. Newt Arnold 3. Arnold Glassman 4. Madison Arnold 5. Maria Arnold 6. Arnold Kent 7. Jason Arnold 8. Monroe Arnold 9. Arnold Brown 10. Arnold Orgolini
Web relevance	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)	1. Arnold Schwarzenegger 2. Arnold Kopelson 3. Tom Arnold 4. John McTiernan (Director of “Last Action Hero”)
MADRank (GRank) for unknown users	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)	1. Arnold Schwarzenegger 2. Arnold Kopelson 3. David Arnold 4. Tom Arnold 5. Arnold Vosloo 6. Arnold Rifkin 7. A. Arnold Gillespie 8. John McTiernan 9. Bonnie Arnold 10. Victor Arnold
MADRank (PRank) for the test user	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)	1. Arnold Schwarzenegger 2. Arnold Kopelson 3. David Arnold 4. Tom Arnold 5. Arnold Vosloo 6. Arnold Rifkin 7. A. Arnold Gillespie 8. John McTiernan 9. Bonnie Arnold 10. Victor Arnold

Table 5: The profile of the test user

Title	Name
Air Force One (1997) (F)	Andy Wachowski (A-)
Commando (1985) (C+)	Steven Spielberg (A+)
Hulk (2003) (C-)	Harrison Ford (B+)
Lord of the Rings: The Fellowship of the Ring (2001) (A)	Keanu Reeves (B)
Lord of the Rings: The Return of the King (2003) (A)	Robert De Niro (A+)
Matrix (1999) (A+)	Tom Hanks (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+)	

eral items including "Terminator (1984)" and "Terminator 2: Judgment Day (1991)" are boosted in the GRank due to their higher item authorities while "Arnold Schwarzenneger DVD 2-Pack - The Sixth Day/The Last Action Hero(2003)" disappears in the top 10 titles due to its low global item authority. In the PRank results, "Terminator (1984)" and "Terminator 2: Judgment Day (1991)" are boosted more since the test user has rated both items higher. Also the rankings of several items including "Total Recall (1990)" and "Terminator 3: Rise of the Machines (2003)" are changed based on the user's expected ratings. By applying item authorities in the ranking function, we believe that search results are significantly improved.

Note that as yet we do not generate personalized item authorities on the name search since we do not collect any user ratings on actors or directors. However, we generate global item authorities on actors and directors using some heuristic methods; We first extract movies for each actor. We only include the movies that the actor played a major role in (within a credit line at 5). Then, we sort movies of each actor based on global item authorities. We sum the global item authorities of the first top 10 movies (the first top 5 movies for directors) and normalized it such that the the global authority of the highest actor and director becomes 13. If an actor has starred less than 10 movies, we only consider them.

We also tested several other heuristic methods, but each method has clear benefits and weaknesses. For example, we also tested a method, which calculating the global authorities of actors based on their average movie authority. However, this method boosts some actors such as "Mark Hamill" (Luke Skywalker) and "Carrie Fisher" (Princess Leia) in the "Star Wars" series, who appear only in few good movies but do not star in many movies later. Note that many popular movie stars such as "Robert De Niro" and "Sean Connery," with some unpopular movies but star in many, will get punished by this kind of average methods. We also tested several variations of the PageRank algorithm to the movie-actor graph and calculated the popularity of movies and actors. However, these approaches do not work well since, unlike a web graph where links imply some kind of recommendation, links in the movie graph do not imply any recommendation but only show some relationship between movies and actors.

MAD6 architecture

The architecture of MAD6 is very simple, which 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 modules can be exchanged with other systems. For example, a system administrator can exchange Yahoo! search engine to AltaVista or Google. Also instead of using the item-based collaborative filtering algorithm, one may use bot-augmented item-based algorithm (Park *et al.* 2005) or ensembles of the MMMF algorithm (DeCoste 2006).

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 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 obtains two inputs from the UI module and Web Analyzer and extracts all informations of items related with the given query, user and the web result. The information contains item contents, global item authorities and user profile information. Then, this information is submitted to the Ranker, which requests the CF Module expected ratings of items for the given user. Then, items are sorted based on the ranking scheme the user has requested.

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; movies and people search for the given query. Search ranking of the two lists can be personalized if a user logs in the system. The user can choose ranking methods such as either global MADRank, personalized MADRank, Web relevance,

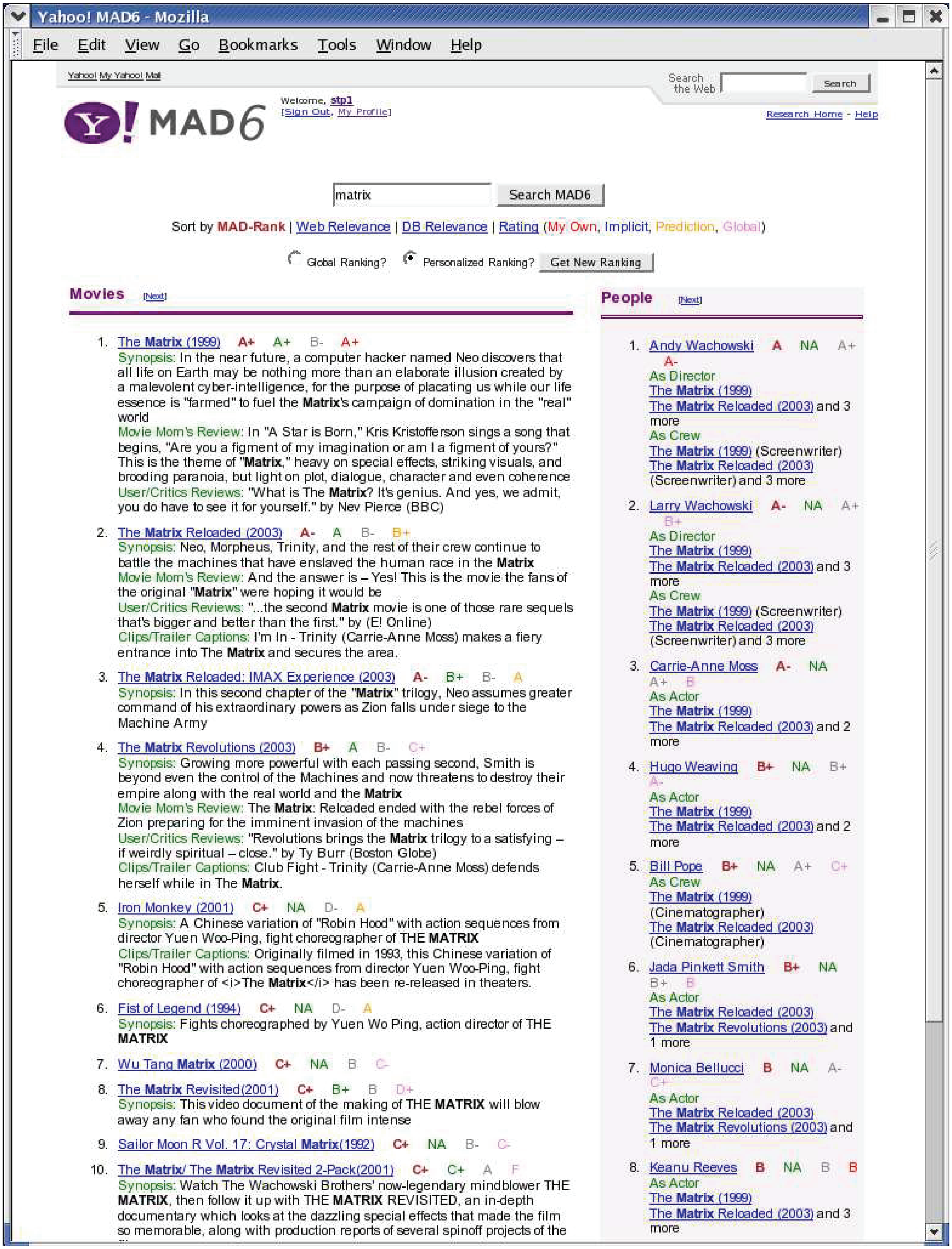

Figure 2: Search Result

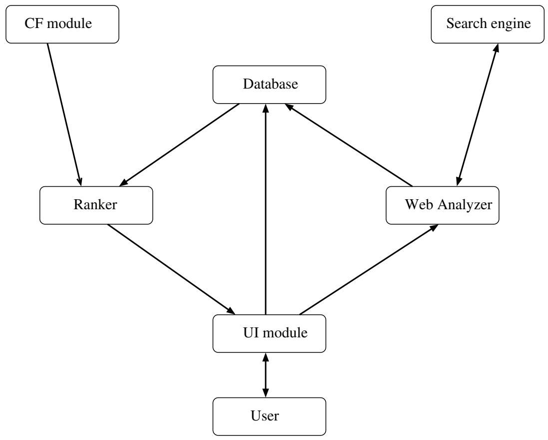

Figure 1: The architecture of MAD6.

DB relevance or item authorities. Each returned item shows ratings based of four methods (MADRank, Web relevance, DB relevance and item authorities) beside of its title and matched fields against the given query. A typical example search result is shown in Figure 2.

In "Item Presentation" pages, MAD6 presents not only information of the presenting item (movies, actors or directors) but also two lists of relevant items based on collaboration of actors and directors or user preferences. For example, when a user clicks on a movie title in a search result, MAD6 presents all information on the movie including poster, ratings, synopsis, release date, reviews and cast and crew information. Also, MAD6 presents a list of relevant movies based on how many cast the movies share with the presenting movie. Moreover, MAD6 presents another list of relevant movies based on *Adjusted Cosine Similarity* (Equation 4) in the user rating space. A typical example of an "Item Presentation" page is shown in Figure 3.

In a "Personal User Profile" page, MAD6 analyzes the activity logs of a user and presents the user personal information such as: (1) What queries has the user submitted most? (2) What movies, actors and directors has the user visited most, either directly or indirectly?[12] (3) What are the user's favorite genres? (4) What movies are mostly recommended for the user? A typical result example of a "Personal User Profile" page is shown in Figure 4.

Future work

One of our future plans for MAD6 is 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 interfaces in the movie domain. We are planing to conduct several online and offline experiments to determine, for example, what percent of real user queries to Yahoo! Movies return meaningful results, and how much improvement can MAD6 provide?

Conclusions

In this paper, we discuss our new ranking method, which combines recommender systems and search tools for better informational search and browsing in E-commerce sites. To visualize the impact of our approach, we have built MAD6, a personalized movie search engine with some unique features. MAD6 seems to provide better search coverage than IMDB and Yahoo! Movies by indexing metadata such as the names of characters, actors, and directors, genres, plots, reviews of users and critics, and awards. MAD6 also provides better search ranking for each user by combining proximities and authorities of the returned items. Even though MAD6 is one application in the movie domain, we believe that our approach is general enough to apply other ecommerce domains including music, travel, shopping and web search.

Acknowledgments

We thank Yahoo! Movies for providing movie and user data.

References

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

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

Basilico, J., and Hofmann, T. 2004a. A joint framework for collaborative and content filtering. In *ACM SIGIR*.

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

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

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

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

DeCoste, D. 2006. Collaborative prediction using ensembles of maximum margin matrix factorization. In *ICML*.

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

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

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

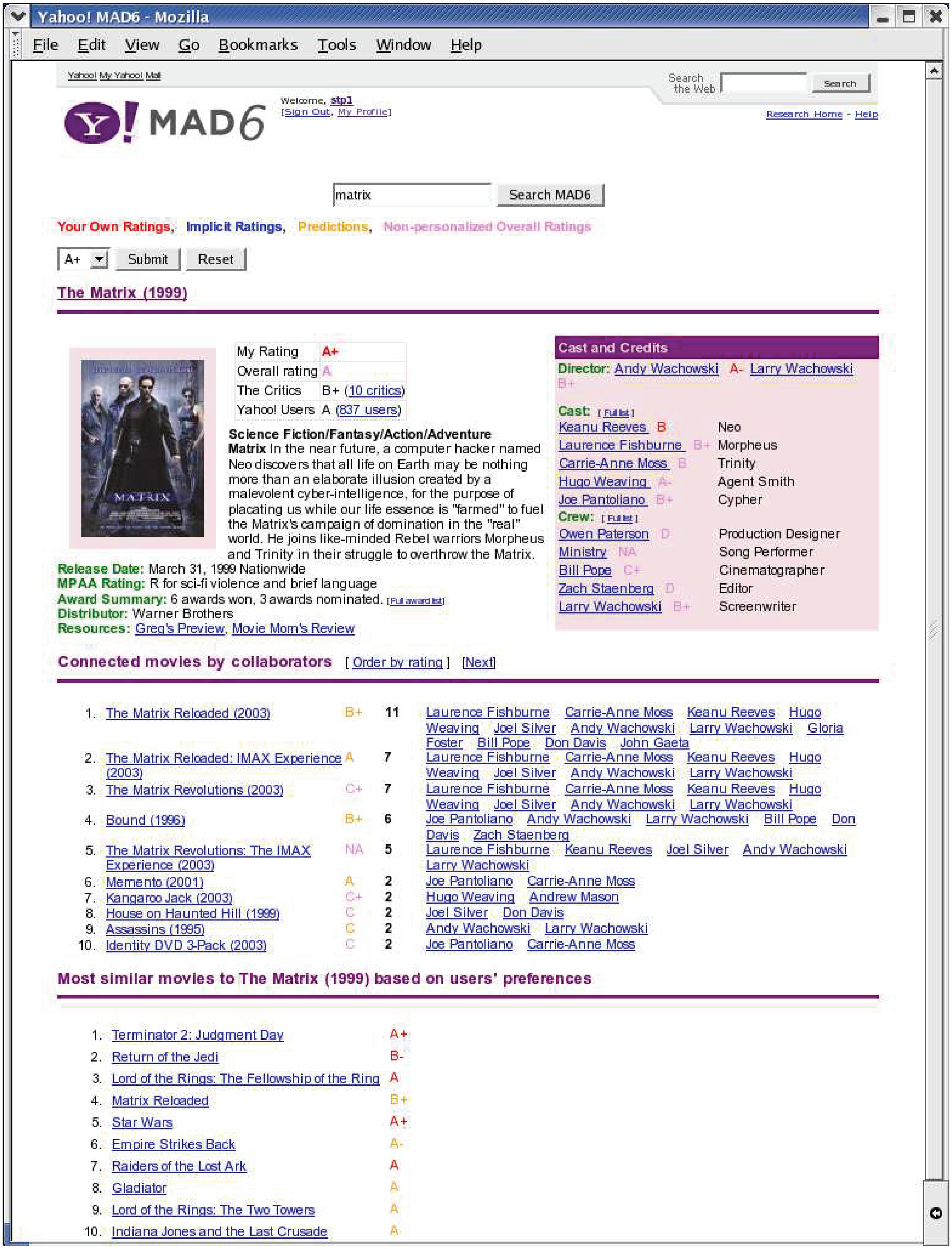

Figure 3: Item Presentation

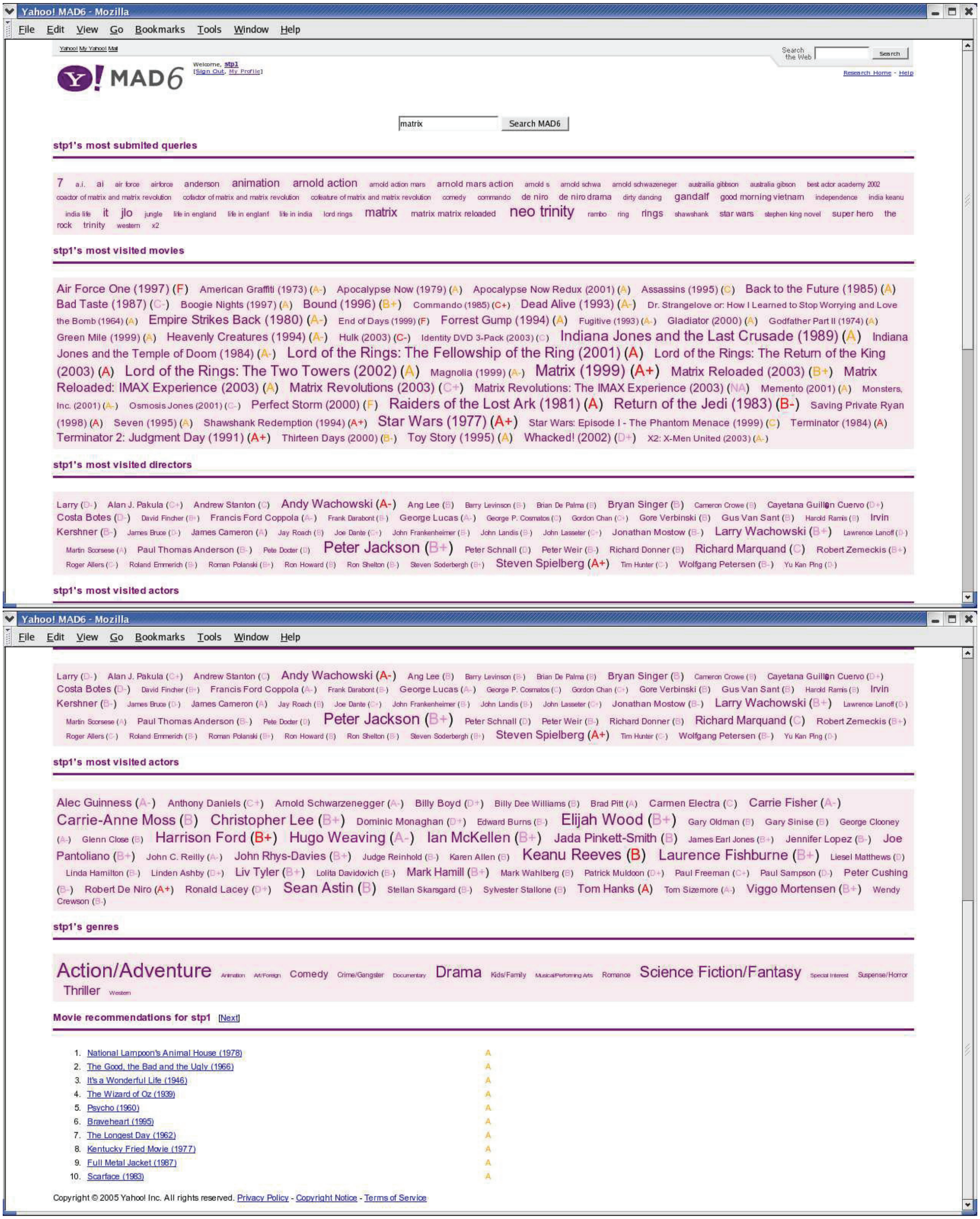

Figure 4: Personal User Profile

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

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

Haveliwala, T. 2002. Topic-sensitive pagerank. In *WWW*, 517–526.

Haveliwala, T. 2003. Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. *IEEE Transactions on Knowledge and Data Engineering* 15(4):784–796.

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

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

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

Karypis, G. 2001. Evaluation of item-based top-n recommendation algorithms. In *CIKM*, 247–254.

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

Konstan, J. A.; Miller, B. N.; Maltz, D.; Gordon, J. L. H. L. R.; and Riedl, J. 1997. GroupLens: applying collaborative filtering to Usenet news. *Communications of the ACM* 40(3):77–87.

Marlin, B., and Zemel, R. 2004. The multiple multiplicative factor model for collaborative filtering. In *ICML*.

McNee, S.; Lam, S.; Konstan, J.; and Riedl, J. 2003. Interfaces for eliciting new user preferences in recommender systems. In *UM*, 178–188.

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

Miller, B. N.; Riedl, J. T.; and Konstan, J. A. 1997. Experience with grouplens: Making usenet useful again. In *USENIX annual technical conference*, 219–231.

Miyahara, K., and Pazzani, M. J. 2000. Collaborative filtering with the simple bayesian classifier. In *PRICAI*, 679–689.

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

Park, S.-T.; Pennock, D. M.; Madani, O.; Good, N.; and DeCoste, D. 2005. Navie filterbots for robust cold-start recommendations. Technical report, Yahoo! Research Labs Technical Report YRL-2005-058.

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

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

Rashid, A.; Albert, I.; Cosley, D.; Lam, S.; Mcnee, S.; Konstan, J.; and Riedl, J. 2002. Getting to know you: Learning new user preferences in recommender systems. In *IUI*, 127–134.

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

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

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

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

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

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

Ungar, L., and Foster, D. 1998. Clustering methods for collaborative filtering. In *Workshop on Recommendation Systems at AAAI*.

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