A Stereotypes-Based Hybrid Recommender System for Media Items

Guy Shani and **Amnon Meisles** and **Yan Gleyzer**
Department of Computer Science
Ben Gurion University, Israel
{shanigu,am,ygleyzer}@cs.bgu.ac.il

Lior Rokach and **David Ben-Shimon**
Department of Computer Science, Department of Information Systems Engineering
Ben Gurion University, Israel
{liorrk,dudibs}@bgu.ac.il

Abstract

Many Recommender Systems use either Collaborative Filtering (CF) or Content-Based (CB) techniques to receive recommendations for products. Both approaches have advantages and weaknesses. Combining the two approaches together can overcome most weaknesses. However, most hybrid systems combine the two methods in an ad-hoc manner.

In this paper we present an hybrid approach for recommendations, where a user profile is a weighted combination of user stereotypes, created automatically through a clustering process. Each stereotype is defined by an ontology of item attributes. Our approach provides good recommendations for items that were rated in the past and is also able to handle new items that were never observed by the system.

Our algorithm is implemented in a commercial system for recommending media items. The system is envisioned to function as personalized media (audio, video, print) service within mobile phones, online media portals, sling boxes, etc. It is currently under development within Deutsche Telekom Laboratories - Innovations of Integrated Communication projects.

Introduction

Recommender Systems — systems that recommend items to users — can be found in many modern web sites for various applications such as helping users find web pages that interest them, recommending products to customers in e-commerce sites, recommending TV programs to users of interactive TV and showing personalized advertisements.

There are two dominating approaches (see e.g. Montaner (Montaner *et al.* 2003)) to creating recommendation systems. The Collaborative Filtering (CF) approach considers the recommended items only by a unique identifier and recommends items that were purchased together, ignoring any attribute of the item. Content-Based (CB) recommendations are generated based on an item profile — a set of attributes of an item — discarding purchase information. Each of these methods has its pros and cons but it seems that a hybrid approach can overcome most of the disadvantages of the two methods.

Modern systems avoid querying the database (of either user-item ratings of item or item descriptions) directly and adopt a statistical model (e.g. Decision Tree, SVD matrix, Dependency Network) to allow scaling up to millions of users and items. Stereotypes (also known as communities) are a way to define an abstract user that has general properties similar to a set (community) of real users. Stereotypes are used in recommender systems for varying purposes, ranging from initial user profile creation to generating recommendations (Montaner *et al.* 2003).

All methods use some type of a user profile (or user model) for recommendation. CF systems usually maintain a vector of rated items while CB systems maintain a rated set of item attributes. We suggest creating a set of stereotype content-based profiles and using an affinity vector of stereotypes as the user profile.

In this paper we present a recommendation system, called MediaScout, that is currently being implemented for Deutsche Telekom Laboratories and intended to be used in mobile phones in Germany for recommending media items such as clips and movie trailers for viewing over the cellular phone. We therefore use media data as examples in this paper but note that our system can be used for other purposes as well. The system is designed to allow mobile, portal or sling box users to obtain most appropriate content (both based on user preferences). As an example, MediaScout provides an *entertain me* option for a user (e.g., waiting for a bus), which when selected delivers content that fits user's personal preferences through her mobile phone.

We suggest to classify new users to clusters through an interactive questionnaire, generated automatically from the stereotypes after each update. Existing users are automatically classified to new stereotypes through the update process and do not need to undergo the questionnaire again.

Recommender Systems

With the explosion of data available online, recommender systems became very popular. While there are many types of recommender systems ranging from manually predefined un-personalized recommendations to fully automatic general purpose recommendation engines, two dominating approaches have emerged - Collaborative Filtering and Con-

tent Based recommendations[1].

Collaborative Filtering

Collaborative filtering stems from the idea that people looking for recommendations often ask for the advise of friends. Over the internet the population that can supply advises is very large. The problem hence shifts into identifying what part of this population is relevant for the current user.

CF methods identify similarity between users based on items they have rated and recommend new items similar users have liked. CF algorithms vary by the method they use to identify similar users. Originally Nearest-Neighbor approaches based on the Pearson Correlation, computing similarity between users directly over the database of user-item ratings were implemented. Modern systems tend to learn some statistical model from the database and then use it for recommending previously rated items to a new audience. Model-based approaches usually sacrifice some accuracy in favor of a rapid recommendation generation process (Breese *et al.* 1998), better scaling up to modern applications.

The main advantage of CF is that it is independent of the specification of the item and can therefore provide recommendations for complex items which are very different yet are often used together. The major drawback of this approach is the inability to create good recommendations for new users that have not yet rated many items, and for new items that were not rated by many users (known as the cold-start problem).

Content-Based recommendation

The ideas of content-based (CB) recommendations originate in the field of information filtering, where documents are searched given some analysis of their text. Items are hence defined by a set of features or attributes. Such systems define a user using preferences over this set of features, and obtain recommendations by matching user profiles and item profiles looking for best matches. Some researchers (Montaner *et al.* 2003) separate methods that learn preferred attributes from rated items (called content-based) from methods that ask the user to specify her preferences over item attributes (called demographic filtering), but we refer to all methods that recommend based on item attribute preferences as content-based recommendation.

Content-based approaches rarely learn statistical models and usually match user profiles and item profiles directly. User and item profiles are very sensitive to profile definitions — which attributes are relevant and which attributes should be ignored. It is also difficult to create an initial profile of the user, specifying the interests and preferences of the user; Users are reluctant to provide thorough descriptions of the things they like and do not like. It is also possible that users are unaware of their preferences. For example, a user cannot know whether she likes an actor she never seen. In fact the acquisition of user preferences is usually considered a bottleneck for the practical use of these systems. Content-based recommendations may also result in very expected items and may not be able to direct the user towards items she is unaware of but may like.

Nevertheless, CB systems can easily provide valid recommendations to new users, assuming that their profile is specified, even if they never used the system before. CB engines can provide recommendations for new items that were never rated before based on the item description and are therefore very useful in environments where new items are constantly added.

Hybrid approaches

The disadvantages CF and CB can be reduced by combining the two approaches into a hybrid method (Burke 2002). Many hybrid approaches use two recommendation algorithms and combine their results in some manner, such as combining the results by their relevance, mixing the output of the two algorithms, switching from CB into CF once the cold-start phase is over, or using the output of one algorithm as input to the second algorithm.

For example, the PVT system (Smyth and Cotter 1999) that operates in a similar setting of online recommendations to TV shows, maintains content-based user profiles and operates two distinct recommendation engines. The CB engine recommends based on program similarity to the programs the user has liked so far, and the CF engine finds the k nearest neighbors based on user profile similarity and recommends programs that the neighbors liked. The final list of recommendations is then created by combining the two lists.

It seems that a more appropriate combination would be to create an algorithm that is by nature a hybrid of CF and CB, not an ad-hoc combination of two independent algorithms.

Stereotypes

Modeling users by stereotypes (or communities) is a well studied concept (Rich 1998). Stereotypes are a generalization of users — an abstract user entity that provides a general description for a set of similar users (a community).

In CF systems stereotypes are described by a set of ratings over items, and user similarity can be identified by their affinity to various stereotypes. In CB systems stereotypes are a set of preferences over item attributes, and users can belong to a single stereotype (Rich 1998) or to multiple stereotypes (Orwant 1995). Recommendations are computed based on the stereotype and then normalized given the user affinity to a stereotype.

Feedback

In order to adapt and refine recommendations to changes in user tastes, most recommender systems rely on some mechanism of feedback from users. Feedback is usually in the form of a rating over an item that can be either numeric (on a scale of, e.g., 1 to 5) or binary (like/dislike).

As users are usually reluctant to rate items explicitly, some research focused on obtaining implicit ratings — estimating the user ratings through her observable operations. For example, in the domain of web browsing, if the user scrolled down the article, or clicked on a link inside the article, then we can assume that the article was useful for her. If

[1]Some researchers (e.g. (Burke 2002)) further divide these classes, but we restrict ourselves to the definitions below.

the user, however, only read the title and then went back to the former page, we can assume that the web page was not useful.

MediaScout

This paper presents the MediaScout system, designed to deliver media content recommendations over mobile phones. The system uses a stereotype approach combining elements from both content-based and collaborative filtering approaches. We explain below how the system is constructed, how new users are introduced to the system, how recommendations are generated and how to update the model.

Stereotype Model

We take the content-based approach here, defining an ontology over media items, defined by an expert in the field. It is reasonable to assume that an expert will be able to identify the key features relevant for people when they choose which movie to see. A media item profile is an instantiation of this ontology, and a stereotype profile assigns relevance values for various attribute values of the ontology. For example, a movie profile may have Bruce Willis and Samuel L. Jackson as actors, and a stereotype profile may assign to Bruce Willis as an actor the value 0.97 and to Mel Gibson as an actor the value 0.85 while assigning to Mel Gibson as a director the value 0.67.

Receiving recommendations for stereotypes can be done by matching item profiles with the stereotype profile, resulting in relevance values over media items.

A user in our domain is modeled by an affinity list of stereotypes. A user may belong for example to one stereotype with relevance 0.8 and to another stereotype with relevance 0.7.

Initialization

Our initial stereotypes are manually defined by an expert. An expert in the field of movies is able to identify several types of movie watchers, such as people who like action movies and people who prefer Sci-Fi. Identifying the relevant actors, directors and other attributes of these stereotypes will also be done by the expert.

When a new user registers into the system we need to create an affinity vector of stereotypes for her. Research in the area has mainly focused on using a set of examples (e.g. a number of movies the user likes) or through a form specifying the user interests. Such approaches are problematic — while rating observed movies is a painless process, using only a set of rated movies can cause the system to later recommend only movies similar to the ones the user rated. Asking users to fill lengthy forms is usually considered a tedious chore and users tend to either avoid it or answer questions arbitrarily (e.g. always picking the first answer).

Smyth and Cotter (Smyth and Cotter 1999), for example, request a new user to fill a form specifying the features she likes and programs she likes and dislikes. They note that users were willing to fill the easier parts of the part but reluctant to look for programs within the lists.

Methods that ask the user to specify her preferences over item attributes are also known as preference elicitation or preference based search. Viappiani et al. (Paolo Viappiani and Evaluating 2006) describes a preference elicitation method and an example-based critiquing, that avoids the problems of preference fluency, domain knowledge and user effort.

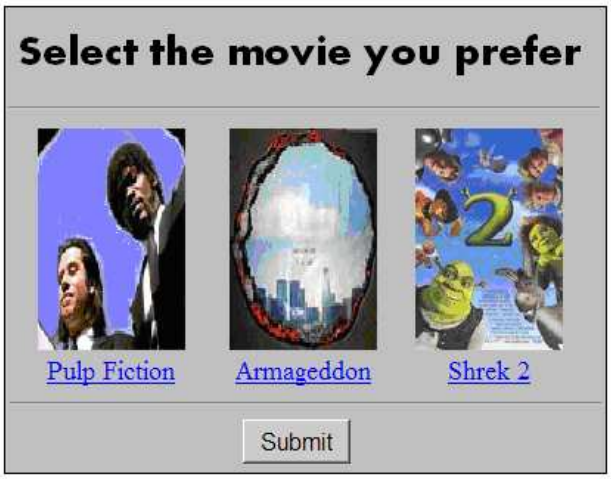


Figure 1: MediaScout questionnaire example.

We propose to combine these two approaches by asking the user a set of simple questions, such as whether she likes an actor, picking a favored movie out of a set of 2 or 3 movies. An example of such a question can be seen in Figure 1. We choose an anytime approach — the user may answer as many questions as she likes. Whenever the user stops answering questions we have some classification of the user into stereotypes. The advantages of the anytime approach have been noted in the past (Pearl Pu and Torrens.).

Our approach is based on a decision tree, with questions at the decision nodes. Each node, both leaves and inner nodes, is assigned an affinity vector of stereotypes. If the user does not wish to answer more questions, the current affinity vector will be assigned to her. The more questions the user answers the more specific her affinity vector becomes.

Recommendations

Once the system obtains a user profile definition in the form of an affinity vector, we can generate recommendations for this user, based on the relevant stereotypes.

First, we need to compute recommendations for the stereotypes. As a stereotype describes a content-based profile, explicitly stating preferences over the possible values of item attributes, we activate a matching engine that computes the relevance of a media item to a stereotype. As the number of stereotypes is not expected to be too high, these lists can be persistent in the database. Moreover, it is expected that many items will have low relevance to stereotypes so the lists can be truncated after some threshold.

Once a request for a recommendation for user u with

affinity vector v is received, we compute the relevance of media item i to user u as follows:

$$relevance(i,u) = \sum_{s \in stereotypes} v(s) relevance(i,s) \quad (1)$$

where $relevance(i,s)$ is the persisted relevance of item i to stereotype s. Note that this process is much faster than matching each user with all items in the database using the matching engine, and therefore can scale up much better.

We would also like to surprise the user sometimes with unexpected content. It is useful to try and explore the user preferences by presenting new media content that the user is unaware of, as many users cannot accurately define what they like (e.g. (ten Hagen *et al.* 2003; Ziegler *et al.* 2005)). Presenting to the user a list of 5 recommendations we can both maintain a high level of acceptance and explore, by always fixing the first 3 recommendations to the most likely items as predicted by Equation 1, and exploring over the last 2 items only. Exploration is done over two different axes — using an ϵ-greedy exploration we select items that have lower relevance given the relevance list. More exploration is done by sometimes randomly boosting the relevance of a random stereotype in the relevance equation. Figure 1a illustrates the list of recommendations that the user gets when she presses the "EntertainMe" button.

Feedbacks

Our system supports both positive and negative feedbacks. As we believe that users find it easier to specify binary, like/dislike feedbacks rather than numeric values, we use binary feedback, but our system can be easily adapted for numeric feedbacks as well.

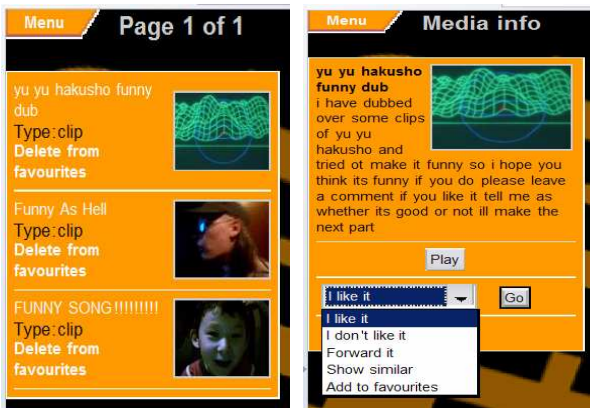
We use two different types of feedbacks — explicit and implicit ratings. For explicit ratings the user can select while watching a media item whether she likes or dislikes it. When a user is presented with a list of 5 items and selects the 3rd item we assume that she did not like the first two items, and notify the system of a negative response for the first two items. Our media content domain uses streaming technology to show media content to users. The server is therefore aware whether the user watched the item fully or decided to stop after a few seconds. If the user decided to stop watching after a few seconds we assume that she did not like the media item and the system is notified of a negative rating.

After a user requests the MediaScout for recommendations, the system displays a list of recommendations. When the user selects a recommendation she may view it (pressing the "play" button) and provide positive or negative feedback using the the combobox below (see Figure 1b).

The gathered feedbacks (implicit and explicit) are used for both model update methods we discuss in the next section.

Model Update

In most environments the relevant items list for a user need to be updated every so often due to the insertion of new items, changes in the information we have over the user (through feedback for example) and general new trends of user tastes. Our system implements three update phases designed to refine the stereotype model.



(a) (b)

Figure 2: MediaScout recommendation list.

Affinity vector update A rapid online update, executed after each feedback received from the user.

When a feedback from a user (either positive or negative) is received, we attempt to update the affinity vector accordingly. We look for the rated item in the relevance lists of the stereotypes. If an item rated positively is relevant to a stereotype, we boost the relevance of the stereotype to the user. If an item was rated negatively by the user, we lower the relevance of stereotypes that recommended it to the user. We use a decaying λ learning factor so that the user affinity vector will eventually converge. λ is reset once the stereotype model is rebuilt (see below).

Recommendation lists recomputation Due to the addition of new items into the database, it is required to recompute the persisted recommendation lists of the stereotypes. We compute new relevance values for new items, and merge them into the persisted lists of recommendations. As we prefer to recommend to the user new items, we also decrease the relevance of items currently in the lists by a constant factor (known as 'aging' the items). Note that recomputation of existing item relevance is unneeded.

The frequency of this recomputation depends on the appearance of new items in the database. In our case it is sufficient to execute the recomputation only once a day.

Stereotype model reconstruction As the initial stereotype model is created by an expert and the initial user affinity vectors are manually created through a questionnaire, we introduce an automatic stereotype discovering phase designed to find new stereotypes automatically and compute new affinity vectors for users. This automatic construction ignores the current user model of affinity vectors of stereotypes.

To create new stereotypes we use a clustering algorithm. Cluster-analysis is a process that receives as input a set of objects, user profiles in our case, and generates clusters (groups) of similar users. The users within a group have high similarity and the users between groups are less similar. The number of clusters can be manually predefined or can be automatically induced by the clustering algorithm.

In hard clustering, the objects are divided into crisp clusters, where each object belongs to exactly one cluster. In soft (fuzzy) clustering, the objects belong to more than one cluster, and associated with each of the objects are membership grades which indicate the degree to which the object belongs to the different clusters. For each cluster, a central stereotype profile (centroid) is generated. The centroid is a weighted average of the users' profiles that belongs to the cluster based on their membership grades.

In this application we use the FCM (Fuzzy C-Means) algorithm (e.g. (Kolen and Hutcheson 2002)), mainly due to its efficient implementation that scales up well to a large number of users. Moreover the FCM uses the probabilistic constraint that the memberships of an object across clusters sum to 1. This constraint suits our approach for weighting stereotypes.

Instead of using the usual distance metric of FCM, we suggest to evaluate the similarity of two users with the Pearson's Correlation metric (see, e.g. (Breese *et al.* 1998)), that compares the similarity of ratings for items (positive and negative). Our experiments show it to produce superior results to standard vector distance metrics such as L_2.
The heavy update phase includes four steps:

- User Representation Construction: for each user we compute a profile similar to the stereotype profiles — a list of preferred values for each possible attribute in the ontology. This profile is automatically computed by observing the list of media items rated positively. For each value of an attribute of an item, we add the value into the user profile. For example, if the user has liked a movie with Bruce Willis, we add Bruce Willis to the list of actors preferred by the user. This can be thought of as *merging* the media items together into a single profile. This profile is used only for the update process, not for computing recommendations for the user.
- Clustering User Profiles: we now use a clustering algorithm to create clusters of similar users using a distance metric over the similarity of user profiles computed in the former step. We use a soft-clustering algorithm, resulting in a list of cluster centroids and for each user, its distances from the centroids of clusters.
- Stereotype Construction: we use each cluster as a new stereotype. To create the stereotype profile, we merge users that are close to the cluster centroid more than a predefined threshold. A cluster (stereotype) profile is defined by the attribute values the users close to its centroid have favored. This merging is weighted by the distance of users from the centroid so that closer users have higher impact.
- Affinity Vector Reconstruction: for each user, we define her new affinity vector as the membership grades to the clusters. Note that these grades are already computed by FCM and there is no need for additional computations. The user shall no longer refer to the manually generated stereotypes, only to the new, automatically generated ones.

We still maintain the manually generated stereotypes and use them to define new users. New users will not have affinity to the automatically generated stereotypes until they will undergo a model reconstruction phase. Alternatively one can automatically learn a new decision tree for the questionnaire from the preferences database by using tree induction algorithms.

Discussion

Our system combines features from various approaches to recommendations. Our initial user affinity vector construction is content-based oriented in that we try to identify attributes of the content of the media item (e.g. actors, director, genre) that the user prefers.

Our heavy update incorporates aspects of content-based (CB) and collaborative filtering (CF). Merging media item attributes to create a user profile of preferred attributes of media elements is classic CB, while clustering users together is more CF oriented. When we create the stereotypes through clustering, we average over a number of users. The resulting recommendations are not based only on items the user has seen and liked, but also on items that similar users have seen and liked, which is a CF approach to recommendations.

The task of classifying new users to stereotypes is a major concern for us. We ask users to go through a questionnaire, providing answers to questions to provide information as to their preferences. Our approach is anytime, allowing the user to stop answering questions whenever she chooses. If a user stops anywhere within the questionnaire, we still have a classification of the user to stereotypes.

Our questionnaire is also easy to use; Questions are short and require the user to select one of a handful of items (e.g. a movie), or attribute values (e.g. and actor). Answers are also provided through images of the available options that the user has to click upon.

The disadvantages of the various approaches are solved. The cold start problem is handled using expert knowledge. The difficulty of having the user properly specify her preferences is eased through our interactive any-time questionnaire. Moreover, even if the profile is inaccurate, it will be refined through our light and heavy updates to better reflect the user preferences. The common problem of CB systems that can recommend to the user only items similar to the ones she has seen and liked does not exist here because users also receive recommendations based on data regarding preferences of similar users.

Experimental Evaluation

Unfortunately, we cannot yet report results from our acceptance tests. To examine the predictive power of the proposed algorithm, we run a prediction test comparing our approach to a number of other algorithms to show its capabilities.

Dataset Used

In order to make our results reproducible, we provide here the results over a public domain dataset. We used the MovieLens [2] database, consisting of 5868 users who rated at least

[2] www.movielens.org

20 movies. The total number of movies in the database is 3288. As our approach requires movie content data, such as actors directors and genres, we used the online Movie Database (IMDB[3]) to collect content for our database. Following Breese et al. (Breese *et al.* 1998), we transformed the ratings, originally in the range of 1 to 5 into binary, like-dislike ratings.

We trained our model over 1000 users using the following method: going over all the movies a user liked, we summed the number of times each attribute value (such as a specific actor name) appeared, divided by the popularity of the attribute value. For example, if the user liked 3 movies with Harrison Ford, out of a total of 23 movies with Harrison Ford in our database, then the user's rating for Harrison Ford is $\frac{3}{23}$. Each attribute class (e.g. Actors) ratings for each user were then scaled to be in the $[0..1]$ range.

This process resulted in a set of user profiles, that were then clustered as described in Section . We then computed the recommendations lists for the resulting stereotypes using the matching engine.

Evaluation Measures

For each user that was not included in our training set we used a part of the movies she liked in order to generate a profile as explained above. We then used the same matching engine (used also by the clustering algorithm) to compute the relevance of this user to each of the clusters — resulting in an affinity vector. We computed a set of recommendations for the user based on this affinity vector. To assess the relevance of the resulting recommendations list, we checked the location of the movies the user liked but were not used in the profile construction. A grade to the recommendations list for user a was computed (following (Breese *et al.* 1998)) as follows:

$$R_a = \sum_{i \in I} \frac{1}{2^{i/\alpha}} \quad (2)$$

where I is the set of locations of the test movies the user liked in the recommendations list. The above metric assume that each successive item in the list is less likely to be viewed with an exponential decay. α is the viewing halflife and in this experiment was set to 10. The grade was then divided by R_{max} — the maximal grade, when all test movies appear at the top of the list.

Following Breese et al. we ran three different tests trying to estimate the accuracy of the prediction given the amount of input data. In the first test ("all but 1") a single movie was selected from the movies liked by the test user and held out. The user profile was then generated using the rest of the movies, and we then computed a recommendation list for this user and graded it, based on the location of the missing item in the recommendation list. The two other tests consisted of building a user profile using 5 ("given 5") or 10 ("given 10) movies and grading based on the location of the rest of the movies the user liked in the recommendation lists.

[3]www.imdb.com

Compared Algorithms

We compared our approach to a number of other algorithms. First, we compared our results to Pearson's Correlation — a well known Collaborative Filtering algorithm that is considered to provide the best CF recommendations (see, e.g. (Breese *et al.* 1998)). Pearson's Correlation is used to establish the highest possible score available. We also compared our approach to random recommendations — the lowest possible grade.

Our true competition, however, is versus a straight forward, content based approach. To create such recommendations, we create test stereotypes as explained above, and then compute direct matching between all movies in the database and the profile. Movies in the recommendation list are arranged in decreasing match value.

Comparative Results

Table 1: Predicting MovieLens movies.

Method	All but 1	Given 5	Given 10
Correlation	4.9	30.3	36.7
Stereotype	3.6	25.1	25.5
Profile	1.7	11.0	11.5
Random	0.37	4.96	4.95
Unseen movies			
Stereotype	9.2	24.7	26.7
Profile	3.8	11.1	11.4
Random	1.6	6.8	7.0

Table 1 shows the grades obtained by the Pearson's Correlation (denoted Correlation), our own stereotype-based recommendations (denoted Stereotype) which has arbitrarily used 40 stereotypes, standard content based matching (denoted Profile) and random recommendations (denoted Random) over the MovieLens dataset.

As we can see, while our hybrid approach provides less accurate results than the Correlation algorithm, it is much better than the direct, content-based movie matching against a user profile. In fact, direct matching is closer to random recommendations than to our approach. This clearly demonstrates how data collected over similar users (CF type data) can help us to strengthen content based recommendations.

As our engine uses an hybridized approach, it would be also appropriate to compare it to a recommendation approach that combines a CF algorithm and a CB algorithm. There are a number of ways two recommendation engines can be combined, creating a hybrid (Burke 2002):

- Combined list: create an interleaved list of the two engines.
- Weighted combination: compute the hybrid rating for a movie as a linear combination of the two engines.
- Waterfall: execute one engine, receiving a list of recommended items, order the items in the list using the second engine.

We implemented all approaches, using Correlation as the CF engine and Profile as the CB engine. In all cases, the results

of the hybrid engine were inferior to using Correlation alone, and in most cases (except for the weighted combination) also inferior to our Stereotype approach. A closer look at the actual lists also shows that the correct ratings always came from the Correlation algorithm and that in the best cases the Profile engine simply did not alter the original Correlation list much.

Predicting New Movies

As we mention earlier, a well known problem with CF algorithms is their inability to predict new items that the algorithm was not trained upon. To test whether our algorithm can handle unobserved items using the item contents, we executed a second experiment. We divided the movies in the database into a training set ($3/4$ of the movies — 2466 movies) and a test set (822 movies). For each user in the users test set, we computed its profile using the movies in the movies train set and tried to predict the movies in the movies test set.

In this scenario Correlation cannot predict any missing movie, since it does not contain the test movies in its database. Thus, we compared only the Stereotype, Profile and Random methods.

As the results in Table 1 show, our stereotype approach is much better than the direct matching approach for this scenario too. This provides a strong evidence towards our system capability to handle unobserved items too, avoid the cold start problem.

In the case of unseen items, the hybrid of two recommendation engines cannot help, as the CF engine supplies no results and therefore all the above approaches reduce to only executing the CB engine.

Model Size

The previous results have been obtained by executing the system with 40 stereotypes. We examine the sensitivity of the results to the number of stereotypes. Table 2 presents the results for previously seen movies and previously unseen movies. Each table specify the results obtained for 10, 20, 30 and 40 stereotypes. The results indicate that for previously seen movies the number of stereotypes has a minor effect. For all three cases (All but 1, Given 5 and Given 10) 20 stereotypes provide the best results.

For previously unseen movies one can identify a small but consistent trend. The performance improves as one increases the number of stereotypes. This implies that for recommendation systems with frequent introduction of new items (such in the case of media clips that can be found in YouTube) one should set the number of stereotypes to a relatively large number to achieve higher predictive performance.

Table 2: Using different model sizes.

# Stereotype	All but 1	Given 5	Given 10
10	3.55	24.71	25.22
20	3.76	25.27	25.72
30	3.70	24.88	25.27
40	3.67	25.12	25.53
Unseen movies			
10	8.44	22.68	24.55
20	8.73	23.51	25.39
30	9.03	24.18	26.15
40	9.24	24.71	26.74

Related Work

As was noted by Burke (Burke 2002), most hybrid recommendation systems combine two algorithms, a CB algorithm and a CF algorithm by either filtering the input of one algorithm into the other (Melville *et al.* 2002), executing the two systems in parallel and combining their lists or other forms of combinations. There is only a handful of algorithms that combine features from various approaches together (Balabanovic and Shoham 1997). Schein et al. (Schein *et al.* 2002) also use content information over items in conjunction with user ratings. They learn sets of actors a user prefers, and then try to find a movie that best suites that set of actors.

Our questionnaire approach for initialization dates back to Rich (Rich 1998), but we are unaware of other systems that implement a decision tree based, anytime, interactive questionnaire similar to the one we use. Also, many systems are designed to recommend web pages, where expert data is less easy to mine, and as such, an expert input for initialization is not very common.

Clustering was also used in the past for learning predictive models. Usually, clustering is performed over user or item ratings (Xue *et al.* 2005; Rashid *et al.* 2006) in order to reduce the complexity of computing the neighborhood of users. Li and Kim (Li and Kim 2003) cluster items based on content and then compute user ratings for items based on item similarity.

The system that is perhaps the most similar to our own is the Doppelganger system (Orwant 1995) — a generic user-modeling system that collects various data over users, creates user models in the form of affinity vectors of stereotypes (called communities). The system can be used to generate recommendations and employs similar ideas about stereotypes and clusterings. Doppelganger, however, does not have our questionnaire initialization mechanism although it uses some expert data to create initial communities. It does not use an ontology, as it is considered usable for any type of items, and can therefore supply less focused content-based data and probably needs more information to generate good user models. It also does not have our light update mechanism for rapid online model updates.

Conclusion

MediaScout — a commercial recommender system for recommending media content to users of mobile phones was presented. It combines ideas from various approaches to recommendations such as expert systems, collaborative filtering and content based recommendations, in a single hybrid algorithm. The algorithm exploits the advantages of the various approaches while minimizing their disadvantages.

This paper also explains how new users are introduced to the system through a questionnaire, explaining its automatic

creation and usage.

The system is currently under development for commercial deployment within the Deutsche Telekom Laboratories - Innovations of Integrated Communication projects, and is expected to be used in mobile phones of Deutsche Telekom.

Acknowledgements

Our matching engine, used for matching user profiles and item profiles was developed by Sandra Zilles of DFKI labs. Our ontology definition and item collection was supplied by CognIT a.s. The project is funded and managed by Deutsche Telekom Laboratories at Ben-Gurion University.

References

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

J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI'98*, pages 43–52, 1998.

R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

J. F. Kolen and T. Hutcheson. Reducing the time complexity of the fuzzy c-means algorithm. 10(2):263–267, 2002.

Q. Li and B. Kim. An approach for combining content-based and collaborative filters. In *6th international workshop on Information retrieval with Asian languages*, pages 17–24, 2003.

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

M. Montaner, B. Lpez, and J. L. De La Rosa. A taxonomy of recommender agents on the internet. *Artificial Intelligence Review*, 19:285–330, 2003.

J. Orwant. Heterogeneous learning in the doppelgänger user modeling system. *User Model. User-Adapt. Interact.*, 4(2):107–130, 1995.

Boi Faltings Paolo Viappiani and Pearl Pu. Evaluating. Preference-based search tools: a tale of two approaches. In *AAAI-06*, 2006.

Boi Faltings Pearl Pu and Marc Torrens. User-involved preference elicitation. In *IJCAI'03*, pages 56–63.

A. Rashid, S. Lam, G. Karypis, and J. Riedl. Clustknn: A highly scalable hybrid model and memorybased cf algorithm. In *WEBKDD*, 2006.

E. Rich. User modeling via stereotypes. pages 329–342, 1998.

A. I. Schein, L. H. Ungar A. Popescul, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *SIGIR*, 2002.

B. Smyth and P. Cotter. Surfing the digital wave: Generating personalised TV listings using collaborative, case-based recommendation. *Lecture Notes in Computer Science*, 1650:561–567, 1999.

S. ten Hagen, M. van Someren, and V. Hollink. Exploration/exploitation in adaptive recommender systems. In *EUNITE03*, Oulu, Finland, 2003.

G. Xue, C. Lin, Q. Yang, W. Xi, H. Zeng, Y. Yu, and Z. Chen. Scalable collaborative filtering using cluster-based smoothing. In *SIGIR*, pages 114–121, 2005.

C. Ziegler, S. McNee, J. Konstan, and G Lausen. Improving recommendation lists through topic diversification. In *WWW*, 2005.