**Proceedings of the 2007 IEEE Symposium on
Computational Intelligence and Data Mining (CIDM 2007)**

Establishing User Profiles in the MediaScout Recommender System

Guy Shani*, Lior Rokach†, Amnon Meisles*, Lihi Naamani†, Nischal Piratla‡, and David Ben-Shimon†
*Computer Science Department, Ben Gurion University, Israel
shanigu/am@cs.bgu.ac.il
†Department of Information System Engineering, Ben Gurion University, Israel
liorrk/ln/dudibs@bgu.ac.il
‡Deutsche Telekom AG Laboratories, Berlin, Germany
Nischal.Piratla@telekom.de

***Abstract*— The MediaScout system is envisioned to function as personalized media (audio, video, print) service within mobile phones, online media portals, sling boxes, etc. The MediaScout recommender engine uses a novel stereotype-based recommendation engine. Upon the registration of new users the system must decide how to classify the new users to existing stereotypes. In this paper we present a method to achieve this classification through an anytime, interactive questionnaire, created automatically upon the generation of new stereotypes. A comparative study performed on the IMDB database illustrates the advantages of the new system.**

I. 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 a number of different approaches to recommendations (Montaner [6] presents a good survey on recommendation approaches), yet all methods use some type of a user profile (or user model) for recommendation. We suggest creating a set of stereotype profiles and using an affinity vector of stereotypes as the user profile. These stereotypes are automatically created during an update process the system undergoes.

In this paper we briefly review the details of our recommendation system; How stereotypes are created and updated, and how recommendations are generated. We mainly concentrate on the task of classifying users to clusters.

We suggest to classify new users to clusters through a questionnaire, that is 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.

Our questionnaire is created as an interactive, easy to use process. At each stage the user is presented with a question and a small number of answers to choose from. The answers are presented as pictures to make the answering process easier to use. The questionnaire is an anytime algorithm, meaning that at each stage the user may choose not to continue, but the system is still able to offer some classification. The more answers the user provides, the more specific the classification becomes.

II. Recommender Systems

With the explosion of data available online, recommender systems [10] became very popular especially in web sites. 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 Content Based recommendations[1].

A. Collaborative Filtering

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

The greatest 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).

B. 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. Researchers sometimes [6] 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),

[1]Some researchers (e.g. [4]) further divide these classes, but we restrict ourselves to the definitions below.

but we refer to all methods that recommend based on item attribute preferences as content-based recommendation.

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.

C. Hybrid approaches

As we see above, the disadvantages of the CF and CB approaches can be solved by combining the two into a hybrid method [4]. 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.

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.

As was noted by Burke [4], 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 [1], executing the two systems in parallel and combining their lists and other combinations. There is only a handful of algorithms that combine features from various approaches together.

Our questionnaire approach for initialization dates back to Rich [11], 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.

The system that is perhaps the most similar to our own is the Doppelganger system [7] — 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.

D. Stereotypes

Modeling users by stereotypes (or communities) is a well studied concept [11]. 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 [11] or to multiple stereotypes [7]. Recommendations are computed based on the stereotype and then normalized given the user affinity to a stereotype.

E. 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 (e.g. [12]) — 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 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.

III. 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 briefly explain below how the system is constructed, how recommendations are generated and how to update the model[2].

We specifically focus on the task of classifying new users to stereotypes through a questionnaire process. We explain how the questionnaire is automatically created and updated, and the process of answering the questionnaire.

A. 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 a stereotype titled "Action" with relevance 0.8 and to a stereotype titled "Comedy" with relevance 0.7.

[2]For farther details on the process of recommendation we refer the reader to [14].

B. 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.

C. Recommendations

As users are defined in the form of an affinity vector over stereotypes, we shall generate recommendations 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:

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

when $relevance(i,s)$ 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.

D. 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.

E. Classifying users to stereotypes

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 boring chore and users tend to either avoid it or answer questions arbitrarily (e.g. always picking the first answer).

We propose to combine these two approaches by asking the user a set of simple questions, such as whether he likes an actor, picking a favored movie out of a set of 2 or 3 movies or demographic questions such as age or occupation. Though it is well-known that users do not like to supply information about themselves we note that, as the service is deployed by a cellular service provider, demographic data is likely to be available to the system anyhow. 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. Note that the advantages of anytime approach have been already studied in the past [8].

Our anytime approach is based on a decision tree, with questions at the decision nodes. Each node, both leaves and inner nodes, is also assigned an affinity vector of stereotypes, so 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, but even without answering any question we can still have some affinity vector under the assumption, for example, that most of the users of such systems tend to be teenagers. Figure 1 illustrates a possible decision tree of questions.

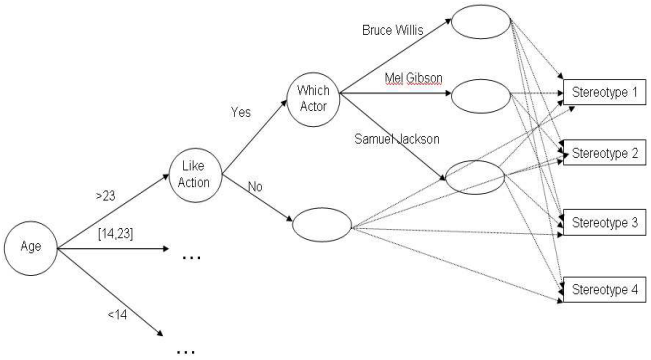

Fig. 1. An illustration of the MediaScout questionnaire decision tree. Round nodes are question nodes, ellipses are terminal nodes. Dotted edges represent the affinity vectors. Affinity vectors were illustrated only for a fraction of the nodes though every node has an affinity vector.

Methods that ask the user to specify her preferences over item attributes are also known as preference elicitaiton or preference based search. Viappiani et al. [16] describes a preference elicitation method and an example-based critiquing, that avoids the problems of preference fluency, domain knowledge and user effort.

F. 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.

Affinity vector update After each feedback received by the system we refine the affinity vector for the user. Given a positive rating for item i we strengthen the weight of stereotypes that favored i and lower the weight of stereotypes that rated i low.

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.

Stereotype model reconstruction As the initial stereotype model is created by an expert and the initial user affinity vectors are 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, so that 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 (see, e.g. [2]), 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. [5]), 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.

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.

Model reconstruction is not intended to be executed often. It is needed only when much data have been gathered through feedbacks, or when we detect a shift in trends in user preferences.

G. Automatic Creation of Hierarchical Questionnaire

When a new stereotype model is reconstructed, one should also create a new hierarchical questionnaire in order to be able to assign new subscribers that join the service to the new stereotype model. In this paper we suggest using a supervised inducer for decision trees. In order to use an inducer, a training set should be created and provided to the algorithm. The target attribute in this training set is the stereotype belonging. In order to facilitate the filling of the questionnaire, we suggest that each input attribute in the training set will refer to a pair of media items, for which the user will be requested to provide her preferences.

Potentially any pair of media items can be used as an input attributes. However as there are many pairs, using all pairs is not practice. Thus we suggest identifying for any pair of stereotypes (s,t), two popular media items that most differentiate between these two stereotypes. For this purpose we suggest to select to the pair of items (i,j) for which term: $relevance(i,s) - relevance(i,t) + relevance(j,t) -$

$relevance(j, s)$ is maximized. Totally we have $\frac{k \cdot (k-1)}{2}$ pairs of media items (questions), where k is the number of stereotypes. Note that all these input attributes are binary (where "1" indicating if item i is preferred on item j, and "0" indicate the opposite).

The next phase is to create instances for the training set. Every instance (row) in the training set refers to the bias of a certain user toward a certain stereotype. The weight of every instance is set to the corresponded entry in the user's affinity vector. The binary values of the input attributes of each user are either known explicitly (based on her previous rating) or inexplicitly (based on her affinity vector).

A different approach for creating the training set, is to have every instance in the training set refers to a movie (and not a user). Moreover the input attributes refer to the movies features: Actors, Directors, etc. For example the feature "Mel Gibson" is a binary input attribute which gets the value "1" if "Mel Gibson" participate in that specific movie. As in the previous case the target attribute is the stereotype assignment.

The final stage is to induce the decision tree. For this purpose we need to a decision tree inducer which can take into consideration the instances weights and which is capable to provide probabilistic classification (namely affinity vector is attached to every node in the tree). We find the well-known C4.5 algorithm [9] suitable for this purpose.

IV. Experimental Evaluation

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.

A. Dataset Used

In order to make our results reproducible, we provide here the results over a public domain dataset. We used the MovieLens [3] database, consisting of 5868 users who rated at least 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[4]) to collect content for our database. Following Breese et al. [3], 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 III-F. We then computed the recommendations lists for the resulting stereotypes using the matching engine.

[3]www.movielens.org

[4]www.imdb.com

B. 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 [3]) as follows:

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

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. Because the number of movies is

C. 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. [3]). 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.

D. Comparative Results

Table I 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)

TABLE I
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

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.

E. 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 I 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.

F. 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 II 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 II
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

G. Hierarchical Questionnaire Evaluation

In this section we examine the ability of the decision tree to classify new users. We traced each user through the decision tree, and compared the classification of the user to clusters to the affinity vector generated by the clustering process. Table III specifies the Euclidean distances MSE of the predicted affinity vector for various training set sizes.

We used four different cases: tracing the tree to a leaf (denoted "All"), tracing the tree one level above a leaf ("All but one"), tracing the tree three levels from the root ("First three") and classifying without answering any questions ("A priori").

As expected the MSE decreases when the size of the training set is increased and when the user provides more information. It is encouraging to note that by providing the answers for the first three questions, the MSE decreases by 43%.

TABLE III
EVALUATING QUESTIONNAIRE CLASSIFICATION PERFORMANCE.

Case	100	200	500	1000
All	0.15	0.12	0.09	0.08
All but one	0.23	0.18	0.11	0.09
First three	0.25	0.22	0.17	0.16
A priori	0.29	0.27	0.28	0.28

V. CONCLUSION AND FUTURE WORK

This paper presents a commercial recommender system for recommending media content, such as movie trailers and clips, to users of mobile phones. It uses an approach that combines ideas from various approaches to recommendations such as expert systems, collaborative filtering and content based recommendations, in a single hybrid algorithm, exploiting the advantages of the various approaches while minimizing their disadvantages.

This paper focused on the way new users are introduced to the system through a questionnaire. It explains the mechanism of this questionnaire, its creation and update process.

Regarding future work, we intend to extend the experimental study and validate the proposed system on other datasets and estimate its performance by applying ROC curve with cross validation. Moreover, one of the drawbacks of the proposed

method is that the user is forced to choose one branch from seemingly closely related alternatives, e.g., Bruce Willis, Mel Gibson, Samuel Jackson. In future versions, the user will be able to express his ambivalence with respect to the choices.

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.

REFERENCES

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

[2] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York, 1981.

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

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

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

[6] 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.

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

[8] Pearl Pu, Boi Faltings and Marc Torrens. User-Involved Preference Elicitation. In workshop notes of the Workshop on Configuration, the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI'03), pages 56-63, August 2003.

[9] J. R. Quinlan. *C4.5: Programs For Machine Learning*. Morgan Kaufmann, Los Altos, 1993.

[10] P. Resnick and H. R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

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

[12] I. Schwab. How to learn more about users from implicit observations. In *UM '01*, pages 286–288, 2001.

[13] I. R. Teixeira, F. de Carvalho, G. Ramalho, and V. Corruble. Activecp: A method for speeding up user preferences acquisition in collaborative filtering systems. In *SBIA*, pages 237–247, 2002.

[14] L. Rokach, G. Shani, D. Ben-Shimon, A. Meisles, N. Piratla MediaScout - an Interactive Hybrid Recommender System for Mobile Phones. Technical Report, Ben Gurion University, 2006.

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

[16] Paolo Viappiani, Boi Faltings and Pearl Pu. Evaluating Preference-based Search Tools: a Tale of Two Approaches. In Proceedings of the Twenty-first National Conference on Artificial Intelligence (AAAI-06), Boston USA, July 16-20, 2006.