Music Recommendation by Modeling User's Preferred Perspectives of Content, Singer/Genre and Popularity

Zehra Cataltepe and Berna Altinel

Istanbul Technical University

Computer Engineering Department

Ayazaga Campus, Maslak, Sariyer, Istanbul, 34469, Turkey

Final Version, July 12, 2008

Book Chapter for the Book

Collaborative and Social Information Retrieval and Access: Techniques for Improved User Modeling

***Abstract*—As the amount, availability and use of online music increase, music recommendation becomes an important field of research. Collaborative, content-based and case-based recommendation systems and their hybrids have been used for music recommendation. There are already a number of online music recommendation systems. Although specific user information, such as, demographic data, education and origin have been shown to affect music preferences, they are usually not collected by the online music recommendation systems, because users would not like to disclose their personal data. Therefore, user models mostly contain information about which music pieces a user liked and which ones s/he did not and when.**

We introduce two music recommendation algorithms that take into account music content, singer/genre and popularity information. In the entropy-based recommendation algorithm, we decide on the relevant set of content features (perspective) according to which the songs selected by the user can be clustered as compactly as possible. As a compactness measure, we use entropy of the distribution of songs a user listened to in the clustering. The entropy-based recommendation approach enables both a dynamic user model and ability to consider a different subset of features appropriate for the specific user.

In order to improve the performance of this system further, we introduce the content, singer/genre and popularity learning algorithm. In this algorithm, we first evaluate the extent to which content, singer/genre or popularity components could produce successful recommendations on the past songs listened to by the user. The number of songs in the final recommendation list contributed according to each component is chosen according to the recommendation success of each component.

We perform experiments on user session data from a mobile operator. There are 2000 to 500 sessions and of length 5 to 15 songs. Our experiments indicate that the entropy-based recommendation algorithm performs better than simple content-based recommendation. Content,

singer/genre and popularity learning algorithm is the best algorithm we investigated. Both algorithms perform better as the session length increases.

***Keywords*—music recommendation, clustering, entropy, user model, audio features, content-based recommendation, collaborative recommendation, popularity.**

I. BACKGROUND

Widespread use of mp3 players and cell-phones and availability of music on these devices according to user demands, increased the need for more accurate Music Information Retrieval (MIR) Systems. Music recommendation is one of the subtasks of MIR Systems and it involves finding music that suits a personal taste (Typke et.al., 2005). The content search in MIR systems could also be used to identify the music played, for example query-by-humming (Ghias et.al., 1995), to identify suspicious sounds recorded by surveillance equipment, to make content-based video retrieval more accurate by means of incorporating music content, to help theaters and film makers find appropriate sound effects (Typke et.al. 2005), to produce audio notification to individuals or groups (Jung & Heckmann 2006).

Music recommendation tasks could be in the form of recommending a single album/song (Logan 2004) or a series of them as in playlist generation (Aucouturier & Pachet, 2002; Alghoniemy & Tewfik, 2000). In addition to containing interesting songs for the user or the user group, a playlist have to obey certain conditions, such as containing all different songs, having a certain duration, having continuity and progression from one song to the next (Aucouturier & Pachet, 2002). Therefore, playlist generation is a harder task than single music item recommendation.

The songs to recommend could contain the audio or MIDI content, as well as, genre, artist, lyrics and other information. The audience of a music recommendation system could be a single person or a group of people (Baccigalupo & Plaza, 2007; McCarthy et.al. 2006). The audience or the songs could be dynamic or mostly static. Depending on these task and user requirements, different algorithms have to be employed for music recommendation. Yahoo Launch!, Last.FM, Pandora (music genome project), CDNow, Audioscrobbler, iRate, MusicStrands, inDiscover (Celma et.al., 2005) are some of the music recommendation projects.

In this chapter, we first review collaborative, content-based and case-based recommendation systems and their hybrids for music recommendation. We also discuss the user models that have been considered for music recommendation. We then introduce two music recommendation algorithms. In our algorithms, we find the perspective of music such as different subsets of audio features, singer/genre or popularity, which affect the song choice of users most. Then we recommend songs to users based on the perspective selected for that specific user.

The rest of the chapter is organized as follows: Section II includes literature review on collaborative, content-based, case-based recommendation systems, hybrid music recommendation systems and user models for music recommendation. In Section III, we go through the motivation for our entropy-based and learning recommendation approaches. Section IV contains the data set we used, the evaluation criteria to compare recommendation algorithms, details on content, popularity and singer/genre information we use for recommendation, our entropy and learning based recommendation algorithms and experimental results and discussions. Section V concludes the chapter.

II. LITERATURE REVIEW

According to Burke (2002), a recommendation system has three components: the background data, user input data and a recommendation algorithm to combine the background and input data to come up with a recommendation. Based on these three components, Burke (2002) comes up with five different types of recommendation systems: collaborative, content-based, demographic, utility based and knowledge based. Among these, collaborative, content-based and case-based recommendation systems are the most used ones.

In this section, we first give an overview of collaborative, content-based and case-based recommendation systems. Then we give examples of hybrid recommendation systems, mostly music recommendation systems, which combine two or more recommendation schemes. We also discuss user models in music recommendation systems.

A. Collaborative, Content-Based and Case-Based Recommendation

Collaborative recommendation systems take as input the users' ratings and generate recommendations based on items selected by users who previously selected similar items. This recommendation scheme does not use the content of the background data and hence has the advantage of being applicable for various data types from movies, to books to food. When there are a lot of users, it takes a lot of time to compute the similarity between them. Item-based collaborative filtering algorithm analyzed in (Sarwar et.al. 2001) overcomes this problem by finding items that are similar to the items that the user liked previously. They show that item-based collaborative filtering performs better than user-based collaborative filtering. Ringo (Shardanand & Maes, 1995) is one of the well-known collaborative music recommendation systems. Based on the ratings given for various artists by the user, Ringo finds other users who have similar taste and recommends groups or music pieces liked by those similar users. GroupLens is another collaborative recommendation system which was used for filtering of usenet news articles (Resnick et.al., 1994).

Collaborative recommendation systems use a user-item matrix, which contains the ratings given by each user to different items. Users who have similar ratings (similar rows of this matrix) are put into the same user group. Pure collaborative recommendation systems suffer from the cold start (or latency) problem, which means a new item which has not been rated by any user will not be recommended to anybody, until some users rate them. Another problem is if a user is unusual, then there will not be users similar enough to him and he will not be able to get reliable recommendations.

Content-based recommendation systems, use similarities of extracted features of items, in order to come up with recommendations. These systems assume that users tend to like to items similar to the ones they liked in the past and compute similarity according to some features based on the content of items. NewsDude of Billsus & Pazzani (1999) is one of the earlier examples of content-based recommendation systems. The main disadvantages of content-based recommendation systems are the difficulty of expressing content similarities, the time it takes to compute content similarities when there are many of them and finally the diversity problem, the fact that the user may not ever be recommended items which are not like the items he has seen in the past but may have actually liked.

There are also *case-based r*ecommendation systems. A case-based reasoning system stores solutions to old problems. When a new problem arrives, it fetches a problem which is likely to have a similar solution. Based on the user response, i.e. whether the solution is what the user required or not, the system learns the new case also. In case-based recommendation systems, user answers some questions on what s/he likes

and s/he is asked some other questions and based on those answers the best item to recommend is found. Wasabi Personal Shopper (Burke, 1999) is one of the early case-based recommendation systems, where users are recommended wines based on answers they give to questions related to quality, price and other properties of wines. Burke (2000) also mentions Entrée and Recommend.com for restaurant and movie recommendation respectively. All of these recommendation systems are based on the FindMe knowledge-based recommender systems where users are asked questions and they can "tweak" the solution they receive according to the answers they provide. Burke (2000) identifies certain aspects of similarity that are important for the recommendation task at hand, for example, cuisine, price and location for a restaurant. Then he produces retrieval strategies which order the similarity metrics according to their importance, for example: first price, second cuisine, and then location. It is mentioned in (Burke, 2000) that collaborative and case-based recommendation could be combined in order to come up with a better recommendation scheme. Another case-based recommendation system is introduced in (Göker & Thompson, 2000). In this system, user models are used for case-based recommendations. The user models are used not only to increase accuracy, but also to enable the user to reach to information s/he seeks, with as small number of questions as possible.

B. Hybrid Music Recommendation Systems

There have been a lot of work on *hybrid* recommendation systems, where collaborative, content-based or case-based recommendation schemes are used together for a better recommendation scheme which does not have the weaknesses of the original ones (Burke, 2002). A hybrid recommender can be built out of existing ones using a number of different methods. A weighted sum that comes from the votes or scores from the existing algorithms can be used to produce recommendations. The system could switch between base recommenders depending on the amount of data available. An example of a switching system is in (Pazzani, 1999), where they decide on which recommender to use based on each recommender's success on the user's past items. Their approach is similar to our work here based on the incorporation of past session history to decide on the recommender. Instead of deciding on a single recommender, recommendations from several different recommenders could also be presented to the user at the same time (Smyth & Cotter, 2000). Using collaborative information as another type of feature in addition to the content feature and then performing content-based recommendation is another option (Basu et. al., 1998). Cascading, feature augmentation and meta-level combination of recommenders are the other types of hybrid recommedation techniques mentioned in (Burke, 2002).

Among the hybrid recommender systems, (Popescul et.al., 2001) built a hybrid method of content-based and collaborative filtering approaches and extended Hofmann & Puzicha's (1994) aspect model to incorporate three-way co-occurrence data among users, items, and item content. They showed that secondary content information can often be used to overcome sparsity. Experiments on data from the ResearchIndex library of Computer Science publications showed that appropriate mixture models incorporating secondary data produce significantly better quality recommenders than k-nearest neighbor (k-NN). Probabilistic models were also used for recommendation. (Smyth & Cotter, 2000) used content-based and collaborative recommendation techniques to come up with a list of programs a person may want to watch on TV. They found out that collaborative approach always resulted in better recommendations than the content-based approach for this problem. Two different types of information: domain and program preferences are kept in the user profile for collaborative recommendation. (Jin et.al., 2005) used both content and user rating information to produce probability of recommendation for each web page.

Yoshii et.al. (2006) introduced a new hybrid music recommendation system incorporating both content-based and collaborative filtering approaches by means of a Bayesian Network. In this method, the distribution of mel-frequency cepstral coefficients (MFCCs) was modeled as music content.

Representations of user preferences differs between collaborative and content-based methods. The former represents a preference of user as a vector that contains rating scores of all pieces. The latter represents the preference as a set of feature vectors of favorite pieces. In order to build a hybrid recommender system, they used a Bayesian network called a three-way aspect model proposed by Popescul et.al. (2001). Their test results showed that their method outperforms the two conventional methods in terms of recommendation accuracy and artist variety and can reasonably recommend pieces even if they have no ratings.

Chedrawy & Abidi (2006) introduce PRECiSE, a collaborative case-based recommendation system and use it for music playlist recommendation. First an item based collaborative filtering (Linden, 2003 ; Sarwar et.al., 2001) is performed, instead of a single value for item-item similarity, a vector of similarities is used. Each component of this vector represents the similarity of items from a perspective, for example lyrics, tune, band etc. The user chooses the relevant perspectives himself. The context similarity of two items is then computed by taking a weighted and normalized sum of perspective similarities. According to this context similarity, for each item rated by the user the closest items are found and a recommendation list is produced based on those closest items. Although (Chedrawy & Abidi, 2006) report that using more (3 instead of 1) perspectives result in better recommendation performance, we believe that if more and more dimensions are added, they could be useless or even harmful because they introduce noise into the context similarity values. In the second stage of the recommendation process, they use case-based reasoning mediation of past cases. They select cases which are most similar to the current user and based on their appropriateness degree with the user, produce recommendations. A weighted sum of the past similar cases are stored into the case database to represent the current user. According to F1 measure, (Chedrawy & Abidi, 2006) show that they get significantly better results with the addition of the case-based reasoning stage.

Li et.al. (2004, 2005 and 2007) showed that using content in addition to user ratings help with collaborative recommendation's three basic problems related to lack of enough data: user bias, non-association, and cold start problems in capturing accurate similarities among items. They performed their experiments on music and video data sets. They use ring tones for cell phones users as their recommendation items. Li et.al. used ratings by users and item attributes together and produced item clusters. They first clustered items together according to their features. Assuming that the ratings of a user in an item community are distributed according to a Gaussian distribution, they produce pseudo-ratings for items that are not yet rated by a user. Both real and pseudo ratings are used to create item groups. An item which has not yet been rated by a user is rated assuming a Gaussian parametric model for ratings for each user. This produces a solution to the cold start problem. In their work, Li et.al., also used the audio features produced by Marsyas (Tzanetakis & Cook, 2002) and experimented with combinations of different features. In our system, we are able to use the Marsyas feature group which best groups the music taste of a user. Another similarity between our and Li et.al.'s work is the cell-phone user data. They work on ring-tones, whereas we work on the songs downloaded by users so that people who call them can listen to instead of the ring-back tone.

Chen & Chen (2005) used both content and user ratings for MIDI music data and achieved better recommendation when using them both. Wang et.al. (2006) perform experiments on collaborative filtering both in text and music recommendation areas.

C. *User Models in Music Recommendation Systems*

User models contain information about a user. By means of the user model, collaborative recommenders or their hybrids are able to find similar users. In a very simplistic sense, once similar users are found, recommendations can be made based on the items they chose and the items the user chose to in the past.

The user information could be gathered explicitly (for example a survey to get sex, age, education, origin etc., see, for example, (Kuo et.al., 2005)) or implicitly, through observation of the user's behavior in the system and then use of machine learning or knowledge-based techniques. Explicit information indirectly and implicit information can be directly used by a recommendation system. The user model can be useful to increase accuracy of recommendation (Sarwar et.al., 2001) as well as making the recommendation process shorter (Goker & Thomson, 2000).

The user's preferences change over time, therefore the user model needs to be dynamic. In their work, Billsus and Pazzani (1999) use two user models, a short term and a long term model, which are modeled using a k-Nearest Neighbor and Naïve Bayes classifiers respectively. In (Rolland 2001) a user model which changes over time as user's preferences change is suggested. Rolland uses alignment similarity between notes in MIDI files of songs to determine their similarity. (Kumar et.al., 2001) used probabilistic user models in their analysis. In this model users select items from different clusterings with some probability and they select each clustering also with some other probability. The probabilistic framework allows analysis of recommendation algorithms in terms of their behavior for different conditions. The work of (Lekakos & Giaglis, 2007) includes lifestyle segmentation of a user for digital interactive television advertisements. Lifestyle includes external (culture, demographics, social status, etc.) and internal factors (perception, motives, personality, etc.). They start with a recommendation algorithm based on the lifestyle of the user and then enrich the algorithm to become a better performing hybrid. They consider both segment level and user-level personalization of recommendations and suggest different levels of personalization based on the amount of data available for the user.

There have been some work to come up with a common language (see UMIRL (User Modeling for Information Retrieval Language) (Chai & Vercoe, 2000), USERML (User model Markup Language) (Heckmann & Kruger, 2003)), in which different systems may describe the user in this standard format to make the user model sharable and reusable.

According to (Chai & Vercoe, 2000), user models are necessary in music recommendation systems due to a number of reasons. User models are needed to specify some perceptual features, for example, to get a happy music, since it depends on all of them, we may need a tuple: <music, user, context, feature, value> to describe it, because a music piece could sound happy to a user based on where s/he listens to it. Demographic and personality factors (such as age, origin, occupation, socio-economic background, personality factors, gender, musical education) have been shown to affect music preference (Uitdenbogerd & van Schyndel, 2002; Yapriady & Uitdenbogerd, 2005), therefore, whenever they are available, they should be included in the user model.

There are a number of internet radio stations which use music recommendation to come up with good songs for users. Among those, Last.FM [www.last.fm] (Aucouturier & Pachet, 2002) creates a user profile based on immediate user feedback and uses collaborative filtering for users. On Pandora Music [www.pandora.com], users provide their approval or disapproval on the individual song choices which later is taken into account. Slacker [www.slacker.com] also operates very similarly. Due to the fact that most people would be unwilling to share their personal information, these radio stations do not ask users about their education, origin, age, sex, etc.

III. MOTIVATION

We aim to use different sources of information, for example audio content, genre/similarity and popularity, in order to come up with better user models and hence better recommendations, for each user individually. Users tend to make choices based on different aspects of music. For example, while someone may like songs based on whether they contain a fast beat, someone else may like them due to their slow beat. In such a case, using only the beat information about a song, we can tell whether these two users would like a song or not. First of all, for all possible feature groups (beat, mfcc etc.), we produce clusterings of all available songs. For each user we choose the feature set that can be used to cluster the songs the user has listened to as compact as possible. If songs are distributed all over the clusters, then it means that the particular feature set is not suitable for that user. Otherwise, the similarity measure could be used to recommend new songs to the user. As a measure of compactness, we use an entropy (Cover, 2006) criterion. When we recommend a certain number of songs, to a user at a certain time, we recommend a certain percentage of songs based on the content of the songs the user has listened to so far and the remaining songs based on the popularity information about songs (Cataltepe, 2007a).

Different feature groups we consider correspond to perspectives of (Chedrawy & Abidi, 2006) which had to be explicitly selected by the user. Finding out the appropriate feature group in this way also automatically allows for dynamic user modeling as in (Rolland, 2001). We should also note that, traditionally recommendation systems build clusters of songs or users in order to make faster recommendations. However, the clustering we do here has the sole purpose of determining the right feature set for the user.

Combinations of different audio content features have been previously used for music recommendation. Li et.al. (2004) combined different audio features based on the proximity of songs and found out that their combination may result in better recommendation performance. Our approach differs from them in that we use all combinations of subsets of audio features in different number of dimensions. We also incorporate song selections by other users in the system. Li et.al. (2004) performed their experiments on a different real-world music corpus, which has 240 pieces of music and 433 users with 1,150 ratings. Our music corpus has 730 audio files and more than 1,350,000 users. Vignoli & Pauws (2005) also combined different audio features. We differ from them in calculation of timbre, genre and mood. They used all features whereas we used different subsets of features. Performance of different subsets of audio features have been examined in (Logan 2004). Logan found out that for the song set recommendation problem, using MFCC features with k-means clustering, minimum distance is a better measure than median or average. For MIDI songs, features extracted from MIDI features could be used, for example (Chen & Chen, 2005). The framework we use in this chapter can be used for MIDI or audio features.

Entropy has been used within the collaborative filtering framework. Pavlov & Pennock (2002) developed a maximum entropy approach for generating recommendations. In order to minimize the case that the recommendations will cross cluster boundaries and then recommending only within cluster, they addressed sparsity and dimensionality reduction by first clustering items based on user access patterns. They performed experiments on data from ResearchIndex and they showed that their maximum entropy formulation outperforms several competing algorithms in offline tests simulating the recommendation of documents to ResearchIndex users.

Every user may also choose songs based on popularity or singer/genre of the songs, and again the importance they give to these properties may also be different for each user. We introduce a framework that lets us estimate the weight of content, popularity and singer/genre for each user based on his/her listening history and make recommendations based on those weights.

IV. STRUCTURE

A. *DATA SET*

User session data is the most important component of a recommendation system. Although there have been recent attempts to produce publicly accessible audio databases (McKay et.al., 2006), we are not aware of a music recommendation database that contains considerable amount of users, sessions and songs.

In the system we consider, cell-phone users pay for and download songs that will stay active in their accounts for upto six months. When someone calls them, instead of the regular ring-back tone, the caller hears one of the songs downloaded by the user. The ring-tone melody of the mobile phones have been colored and the phone melodies have become a huge market. Subscribers have paid and are still paying for their customized, popular ring tones. There are many operators sharing revenues with the ring tone content providers. Now the market that is created by the ring tones may be expanded to the ring back tones. Traditionally, subscribers listen to the infrastructure based tones before connecting to the other party. The traditional tones are played at the call states of alerting, busy, not reachable, and no answer. In the system from which our data set comes, customers change the traditional ring back tones with the melodies they select. Although the choice of songs downloaded may be affected by the identity of people who call the user, the user is the person who picks the songs not the calling party. On the other hand, the song preferences of a user may also change in time. Since the songs downloaded in the previous part of a session are used for the same purpose as the songs downloaded at the later parts, in other words, the characteristics of the recommendation task does not change, we believe that this is a legitimate music recommendation task. We should also note the fact that the songs downloaded are not gifts for other people.

We are provided with the identity of the songs and the times they are selected for each user. There were sessions containing different number of songs, however, we concentrated only on sessions of length 5, 10 and 15 songs. There were a total of 11398, 1215 and 518 user sessions of length 5, 10 and 15. Due to time limitations, in our experiments, we used 2000, 1000 and 500 of these sessions respectively. There were a total of 730 songs, whose audio features we obtained as described below.

In our recommendation system, no feedback about the recommendation is given to the users and we can not evaluate the live system performance. We evaluate the system performance based on how well we can predict what user selects at a particular point in time.

B. *EVALUATION OF RECOMMENDATION ALGORITHMS*

Many metrics have been proposed to evaluate recommendation algorithms. (Herlocker et.al., 2004) provides a detailed review of evaluation metrics and their suitability for different tasks, number of items and users in the recommendation system and the kind of input available from the user. In terms of the tasks, the task we consider in this chapter is the task of “Find Good Items”. Although, a live evaluation would be a lot more reliable, it was not possible to perform it for the recommendation system proposed, hence the evaluation results are based on user requests that already happened. However, we make sure that the evaluation results are for test items only, we separate the last item in a session for a user and evaluate the performance of the system for that item. Since we have timestamps for each item request, we make sure that we only use information available at the time of recommendation.

In order to evaluate our recommendations, we can not use metrics that rely on multi-valued ratings, since they are not available. In our system, what we really have are, what (Herlocker et.al., 2004) calls “unary” ratings, since we only have a record of items user selected, an item which has not been selected in the past

is not necessarily disliked by the user. The metrics we could have used are precision, recall, F1 metric, ROC (Receiver Operating Characteristic) curve and AUC (Area Under ROC Curve or Swet's A Measure). Precision is the number of selected and relevant items divided by the number of selected items. Recall is the probability that a relevant item is selected and it is computed by dividing the number of relevant and selected items to the number of relevant items. As the number of recommended items increases, recall increases and precision decreases. F1 measure attempts to combine both metrics into one. While F1 measure combines precision and recall for a single recommendation list length, ROC curve provides a curve of precision and recall values for all possible recommendation list lengths. The AUC is the area under an ROC curve and it makes comparison of algorithms' performance easier. If a recommender is able to find the items at the beginning of the recommendation list, it would have higher values of precision at the beginning of the ROC curve and hence a higher AUC value.

In our study, we use percentage of times the correct song was in the list of recommendations returned as an evaluation measure. The same metric has been used in (Logan, 2004) for a number of different recommendation list lengths. We use a recommendation list of length 20 for all algorithms. As the recommendation list gets bigger this evaluation measure would increase. (Logan, 2004) also used a more relaxed metric where she assumed that a recommendation was correct if the composer name was correct. Error rate (which is one minus percentage of times correct) has been used in, for example, Jester joke recommendation system (Goldberg et al., 2001). Yoshii (2006) also used accuracy to evaluate their recommendation system.

C. CONTENT INFORMATION

For each song in our data set, we have the audio files, from which we extract the audio features described below. Singer and genre of each song is also known. Based on the songs listened within a certain number of days, we also obtain a popularity measure for each song.

1) Audio Features

Several features including low-level parameters such as zero-crossing rate, signal bandwidth, spectral centroid, root mean square level, band energy ratio, delta spectrum, psychoacoustic features, MFCC and auditory filter bank temporal envelopes have been employed for audio classification (Uitdenbogerd & van Schyndel, 2002). In our experiments, we obtained the following content-based audio features using Tzanetakis's Marsyas software [opihi.cs.uvic.ca/marsyas] with default parameter settings (Tzanetakis & Cook, 2002).

Timbral Features (MFCC and STFT)

Timbral features are generally used for music-speech discrimination and speech recognition. They differentiate mixture of sounds with the same or similar rhythmic content. In order to extract the timbral features, audio signal is divided into small intervals that can be accepted as stationaryl. The following timbral features are calculated for these small intervals: spectral centroid, spectral rolloff, spectral flux, time domain zero crossing, low energy, mel-frequency cepstral coefficients (MFCC).

Means and variances of the spectral centroid, spectral rolloff, spectral flux, zero crossing (8 features) and low energy (1 feature) results in the 9 dimensional feature vector and it is represented in experimental results using the STFT label. Means and variances of the first five MFCC coefficients yield a 10 dimensional feature vector, which is labeled as MFCC in the experiments.

Rhythmic Content Features (BEAT)

Rhythmic content features characterize the movement of music signals over time and contain such information as the regularity of the rhythm, the beat, the tempo, and the time signature (Tzanetakis & Cook, 2002; Li & Tzanetakis, 2003). The rhythm structure is detected based on the most pronounced periodicities

of the signal. Rhythmic content features are calculated by beat histogram calculation and yield a 6 dimensional feature vector which is represented using BEAT label.

Pitch Content Features (MPITCH)

The melody and harmony information about the music signal is obtained by pitch detection techniques. Although musical genres may not be characterized fully by their pitch content, there are certain patterns that could lead to useful feature vectors (Tzanetakis & Cook, 2002). Pitch content features are calculated by pitch histogram calculation and yield a 5 dimensional feature vector which is labeled as MPITCH in the experimental results.

The following is a list of audio features we use and their length:

- BEAT (6 features)
- STFT (9 features)
- MFCC (10 features)
- MPITCH (5 features)
- ALL (all 30 features above)

Using CLUTO (Karypis, 2002) software and graph clustering option, we obtain 8 different clusterings of all the 730 songs in our database.

We use the Euclidean distance between song features as the distance between songs. If $\underline{x}$ and $\underline{y}$ are audio feature vectors of dimension d for two songs x and y, the distance between x and y is:

$$d_{audio}(x, y) = \left\| \underline{x} - \underline{y} \right\| = \sqrt{\frac{1}{d} \sum_{i=1}^{d} \left(x(i) - y(i) \right)^2} \qquad (1)$$

We also considered the cosine similarity between song features, however did not observe a significant performance difference. We obtained 8 clusterings using the following (*audio*) feature combinations: MFCC, MPITCH, STFT, BEAT, MFCC+MPITCH, STFT+BEAT, MPITCH+BEAT and ALL. We considered all 15 possible feature set combinations, but discarded combinations for which the clustering algorithm can not perform well (i.e. very non-homogenous clusters, many songs outside clusters, etc.).

2) ***Singer/Genre Information***

Singer/genre distance value is calculated according to the 4 level hierarchy presented to the cell-phone users: Turkish/Foreign song, genre, singer and song (Figure 1). Because users are presented the song information in this way, what they select is affected by it. Using singer/genre distance, we wanted to take the effect of this presentation into account. If two songs share the same singer (lowest category) then their singer/genre distance is 0, if they do not share the same highest category, their singer/genre distance is 1. We denote the singer/genre distance between two songs x and y as $d_{\sin ger}(x, y)$.

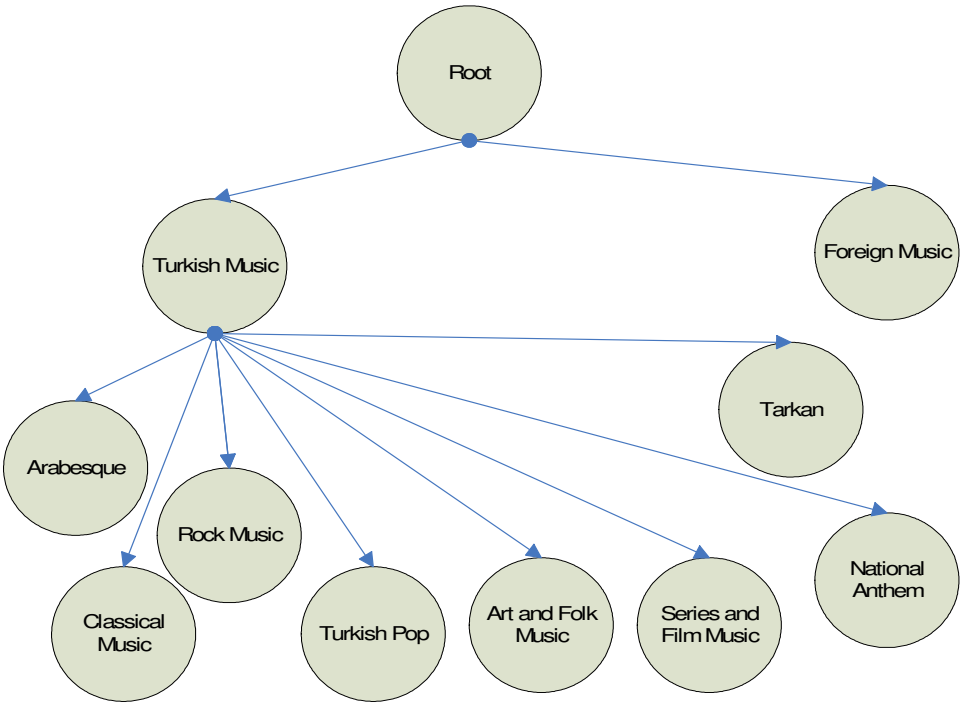

Figure 1. *Some of the singer/genre categories.*

3) ***Popularity Information***

For any day of recommendation, we group songs into popular and non-popular. We compute the popularity ratio as the number of times a song is listened within the last t days divided by the number of times all songs are listened within the last t days. We compute the mean popularity ratio for all songs and group songs whose popularity ratio are below the average as unpopular and the rest as popular. (Ahn, 2006) is another study that uses popularity for recommendation.

We use the popularity matrix in order to compute the popularity values (Table 1). The popularity matrix contains the count of the times a specific song is requested by a user and the count of how many songs were requested among all 730 songs. In order to compute the popularity ratio of a song on a specific day, we take the ratio of these two counts (Table 2).

Table 1. *A sample table showing the number of times each song is listened on a day (Popularity Matrix)*

Day	Bryam Adams-I need Somebody	50 Cent-Just A Little Bit	Jennifer Lopez-Play	Paris Avenue-I Want You	…	All Songs Listened
01.01.2006	3	0	2	1	…	34
02.01.2006	5	1	4	2	…	78
03.01.2006	6	2	12	5	…	101
04.01.2006	11	4	10	7	…	124
05.01.2006	25	0	1	0	…	45
…	…	…	…	…	…	…

Table 2. *A sample table showing the popularity values computed Table 1 above.*

Day	Bryam Adams-I need Somebody	50 Cent-Just A Little Bit	Jennifer Lopez-Play	Paris Avenue-I Want You	…	All Files Listened
01.01.2006	0.088	0	0.058	0.029	…	34
02.01.2006	0.064	0.013	0.051	0.015	…	78
03.01.2006	0.059	0.02	0.119	0.05	…	101
04.01.2006	0.089	0.032	0.08	0.056	…	124
05.01.2006	0.556	0	0.022	0	…	45

D. RECOMMENDATION ALGORITHMS

In this section, we first introduce the notation that will be used for our recommendation algorithms. Then we give the two recommendation algorithms: entropy-based and content, singer/genre, popularity learning based recommendation.

1) Notation

Let $s(i) = [s(i,1), s(i,2),..., s(i,N_i)]$ represent the i'th user session containing N_i songs. $s(i,j)$ represents the j'th song of the i'th session. Each song is represented by means of the 30 dimensional audio feature vector, $x_{i,j} \in R^{30}$ consisting of MFCC, MPITCH, STFT and BEAT features described above**.**

$t(i) = [t(i,1), t(i,2),..., t(i,N_i)]$ is the vector of the times at which the songs in session *s(i)* were chosen.

The recommendation task that we consider is the following: Given the portion of a session excluding the last song, $s(i)^{-} = s(i,1), s(i,2),..., s(i,N_{i-1})$, recommend M songs from among the 730 songs at time $t(i,N_i)$. The value of M needs to be selected carefully, based on the number of items user can examine and the

error level that can be tolerated.

A recommendation consists of $M = \sum_{k=1}^{K} M_k$ songs, where M_k represents the number of songs recommended according to similarity measure k. We consider $K = 8$ different types of information. $k = 1...8$ correspond to the 8 different subsets of audio features MFCC, MPITCH, STFT, BEAT, MFCC+MPITCH, STFT+BEAT, MPITCH+BEAT and ALL.

Below we describe the recommendation algorithms used in this study. The experimental results for each algorithm are given in the Experimental Results section.

2) *ENTROPY-BASED RECOMMENDATION*

When we need to recommend M songs using $s(i)^- = s(i,1), s(i,2), ..., s(i, N_{i-1})$ at time $t(i, N_i)$, for each clustering $k = 1...8$, we compute the number of songs to recommend from this clustering as follows. For the clustering c, we first find the cluster to which each song belongs. Let:

$$p_c = n_c / (N_i - 1) \quad (2)$$

be the ratio songs in session $s(i)^-$ which fall into a cluster c. In equation (2), n_c is the number of songs in $s(i)^-$ assigned to the cluster c, where $c \leq C = 20$ and C is the number of clusters.. The (Shannon) entropy (Cover, 2006) value for this clustering is computed as:

$$H_k = -\sum_{c=1}^{C} p_c \log p_c \quad (3)$$

The number M_k of songs to recommend from clustering k, should decrease as the value of H_k increases. Because a high value of entropy means songs in the session $s(i)^-$ are distributed all over the clustering k. In this study, we use a *discrete scheme* to compute M_k .The clustering whose entropy is minimum is selected as the clustering to which the user belongs, because it is the clustering that can group the songs user has listened to in the best possible way. All M songs are recommended from that clustering. Figure 2 shows an example depiction of the entropy-based recommendation.

When popularity P, where $0 \leq P \leq 1$, is also included in recommendation, first $P*M$ most popular songs at the time of recommendation are recommended. The remaining $(1-P)*M$ are selected according to the entropy-based scheme above.

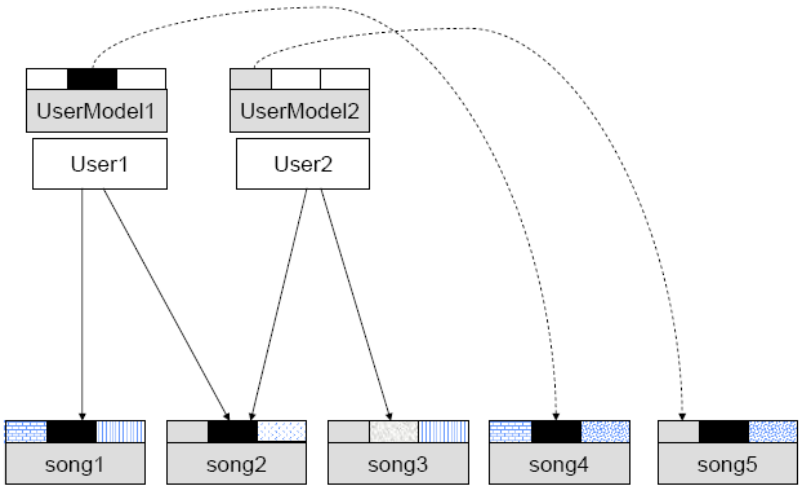

Figure 2. *A depiction of the entropy-based recommendation. Different perspectives/feature groups of songs are shown with small rectangles. User1's songs can be grouped as having common black second property, User2's songs have their first property gray. The second property can be used to select songs for the first user and the first property can be used to select songs for the second user. Hence song4 can be recommended to User1 and song5 can be recommended to User2.*

3) ***CONTENT, POPULARITY AND SINGER/GENRE LEARNING BASED RECOMMENDATION***

Entropy-based recommendation recommends a certain percentage P popular songs for all users in the system. However, just like favoring different sets of audio features, users could show different preferences for songs based on their popularity or singer/genre. In content, popularity and singer/genre learning based recommendation, in addition to user's preferred set of song features based on entropy, we also learn his/her degree of preference for songs based on their popularity and singer/genre. This way, instead of system wide values for P, we can have a different P value per user based on the user's history of songs. We can also incorporate the singer/genre preferences for the specific user.

In this method we consider all three components (content similarity, singer/genre similarity and the popularity) and learn the percentage values for each component. We do the learning as follows: For each $s(i,j) \in s_i^-, j < N_i$, we try to find $s(i,j)$ based on all remaining $N_i - 2$ songs in the session. We use content, singer/genre and popularity components all by themselves for finding song $s(i,j)$ when they give M recommendations. We choose the number of songs to recommend from each of the content, singer/genre and popularity components proportional to their number of successes in recommending item $s(i,j)$.

As seen in Figure 3, each user could be given recommendations from a different recommender based on the success of content, singer/genre (not shown) or popularity recommenders on the past user session data.

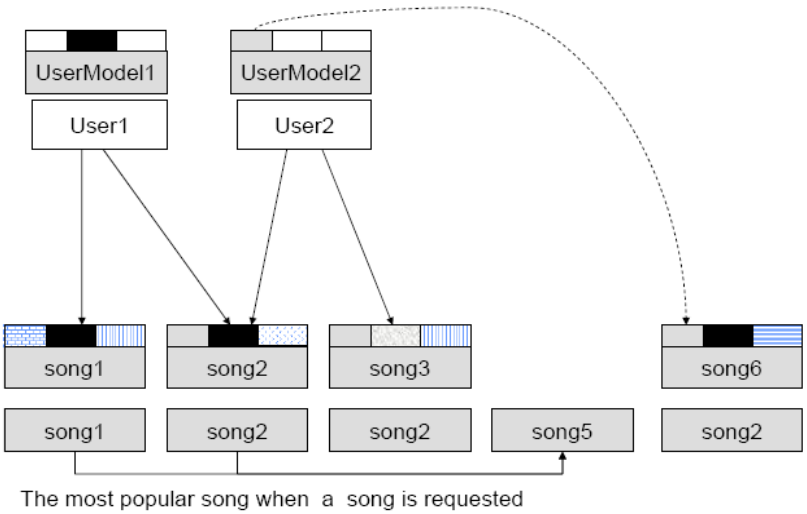

Figure 3. *A depiction of the popularity, singer/genre learning based algorithm. The songs requested by User1 are always the most popular songs therefore the recommendation for that user is song5, which is the most popular song when recommendation is requested. On the other hand, songs listened by user 2 are not always the most popular, therefore, user model 2, not the popular songs, is used to get the recommendation for user 2.*

E. EXPERIMENTAL RESULTS

In Table 3, we give the results of simple recommendation, based on a single set of song features for all users and entropy-based recommendation, which uses the minimum entropy set of features for each specific user.

Results are shown for varying ratio P of recommendations from the popular songs. The number of sessions considered are 2000, 1000 and 500 for sessions of length 15, 10 and 5 respectively. A recommendation is considered successful if the N_i'th song is among the recommended songs. M=20 songs are recommended in all cases, therefore if songs to be placed on the recommendation list were selected randomly without replacement from among the 730 songs, the probability of success would be 20/730=2.74%.

Percentage of songs recommended from among the popular songs at the time of recommendation are shown on the second column. As more popular songs are considered for recommendation, success increases. The remaining songs are recommended using entropy-based method. Column 3 in the table shows the recommendation success when the entropy of clusterings of songs in user history are used to select the best clustering for the user among 8 different clusterings. Columns 4, 5 and 6 shows the recommendation success when only a static set of features and hence clustering (ALL, MPITCH+MFCC, BEAT+STFT) are used. The entropy-based recommendation results in 10 to 62 percent better recommendation success. With increasing the session length the entropy-based method becomes even more successfull, because feature set and hence the clustering valued by the user in selecting a song can be predicted more reliably. When content-based recommendation is done based on only a static set of features, using ALL features results in better recommendation success than using any subset of features.

Table 3. *Recommendation success when 8 clusterings and entropy measure vs. a static single clustering is used.*

Session Length	P = %Popular Recommend	%RecomSuccess Entropy-based	%RecomSuccess All Features	%RecomSuccess MFCC+MPTCH	%RecomSuccess STFT+BEAT
5	20	**21**	19	11	11
5	40	**30**	22	16	13
5	60	**40**	28	14	17
5	80	**44**	33	22	19
10	20	**22**	18	13	13
10	40	**32**	25	16	18
10	60	**41**	27	13	13
10	80	**46**	29	20	17
15	20	**22**	17	8	11
15	40	**33**	21	16	13
15	60	**44**	27	15	15
15	80	**50**	32	17	17

Table 4 compares all the algorithms considered in this study. As seen in the table, content, singer/genre and popularity learning performs best and entropy-based recommendation method follows. Both methods perform better than content-based recommendation based on all the available song features.

Table 4. *Comparison of Recommendation Accuracies of Simple (Using ALL features), Entropy-based, Popularity and Singer/Genre Learning and User Group Learning Algorithms.*

Session Length	%RecomSuccess Simple Recommendation Using ALL features (P=80%)	%RecomSuccess Entropy-based Recommendation (P=80%)	%RecomSuccess Content, Popularity, Singer/Genre Learning
5	33	44	**70**
10	29	46	**71**
15	32	50	**73**

As the session length increases, there is more information available about user's preferences. While simple recommendation has not benefited much from this information, both entropy-based and content, singer/genre and popularity learning based recommendation algorithms were able to get better. When session length is very small, the entropy values computed become less reliable. It could be a good idea to incorporate the session length into the content, singer/genre and popularity learning algorithm.

V. CONCLUSIONS AND FUTURE WORK

In this chapter, we introduced a number of new ideas for music recommendation based on different types of available information. First of all, we introduced a framework that lets us use different subsets (portions) of audio features for each user so that we can do recommendation to a user, based on the most relevant subsets of features for that user. We used the entropy measure to decide on which subset of features to use for a particular user. We then introduced a recommendation algorithm where content, popularity and singer/genre preference for each user are computed and the best performing recommender is selected for that user.

Analyzing the performance of these recommendation systems on a running system and using them not only for music recommendation, but also for web page recommendation are among the future work that we consider.

Acknowledgements

We thank Argela for providing the user session data, G. Tzanetakis for Marsyas software, G. Karypis for Cluto software. We thank Prof. Sule Gunduz-Oguducu of Istanbul Technical University for useful discussions and proofreading. Author Cataltepe would like to thank Dr. Tanju Cataltepe for his continuous support and also proofreading this work. Authors also appreciate anonymous referees' comments which greatly helped improve the quality of this work.

References

Ahn, H. J. (2006). Utilizing Popularity Characteristics for Product Recommendation. *International Journal of Electronic Commerce* / Winter 2006–7, Vol. 11, No. 2, pp. 59–80.

Alghoniemy, M. & Tewfik, A.H. (2000) User-defined Music Sequence Retrieval. *Proceedings of the eighth ACM international conference on Multimedia.* 356 - 358.

Aucouturier, J.J. & F. Pachet, F. (2002) Scaling up Music Playlist Generation. *Proc IEEE Intl Conf on Multimedia Expo*.

Baccigalupo, C. & Plaza, E. (2007) A Case-Based Song Scheduler for Group Customised Radio. *ICCBR* 2007, LNAI 4626, pp. 433–448, 2007.

Billsus, D. & Pazzani, M.J. (1999) A Hybrid User Model for News Story Classification. *Proc. the Seventh International Conference on User modeling,* Banff, Canada, 99 - 108.

Billsus, D. and Pazzani, M. 2000. User Modeling for Adaptive News Access. *User-Modeling and User-Adapted Interaction* 10(2-3), 147-180.

Burke, R. (1999). The Wasabi Personal Shopper: A Case-Based Recommender System. *Proceedings of the 11th Conference on Innovative Applications of Articifial Intelligence.* American Association for Artificial Intelligence. pp. 844-849.

Burke, R. (2000) A Case-Based Approach to Collaborative Filtering. *Advances in Case-Based Reasoning,* pp. 370–379, *5th European Workshop EWCBR 2000.* Springer-Verlag, New York.

Burke, R. (2002). Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, 12: 331-370.

Cataltepe, Z., & Altinel, B. (2007a). Hybrid Music Recommendation Based on Different Dimensions of Audio Content and Entropy Measure. *Proc. of Eusipco 2007 Conference,* Poznan, Poland, September 2007.

Cataltepe, Z., & Altinel, B. (2007b). Hybrid Music Recommendation Based on Adaptive Feature and User Grouping. *Proc. of ISCIS 2007 Conference*, Ankara, Turkey, October 2007.

Celma, O., Ramirez, M., & Herrera, P. (2005). Foafing the Music: A Music Recommendation System Based on RSS Feeds and User Preferences. *Proc. of the International Conference on Music Information Retrieval (ISMIR) 2005.*

Chai, W. & Vercoe, B. (2000) Using User Models in Music Information Retrieval Systems. *Proc. of the International Conference on Music Information Retrieval (ISMIR) 2000.*

Chedrawy, Z., & Abidi, S. S. R. (2006). An Adaptive Personalized Recommendation Strategy Featuring Context Sensitive Content Adaptation, Adaptive Hypermedia and Adaptive Web-Based Systems. *LNCS* Volume 4018/2006, 61-70.

Chen, H.C., & Chen, A.L.P. (2005). A music recommendation system based on music and user grouping. *Journal of Intelligent Information Systems*, Volume 24, Numbers 2-3, 113-132.

Cohen, W., & Fan, W. (2000). Web-collaborative filtering: Recommending music by crawling the Web. *Computer Networks*, vol. 33, no. 1–6, pp.685–698.

Cover, T.M. & Thomas, J. A. (2006). *Elements of Information Theory.* Wiley.

Ghias, A., Logan, J., Chamberlin, D., & Smith, B.C. (1995). Query by humming - musical information retrieval in an audio database. *Proceedings ACM Multimedia.*

Goker, M.H. & Thompson, C.A. (2000) Personalized Conversational Case-Based Recommendation. *Advances in Case-Based Reasoning,* LNCS 1898/2000, 29-82.

Heckmann, D. & Kruger, A. (2003) A user modeling markup language (UserML) for ubiquitous computing. *Lecture Notes in Artificial Intelligence* 2702 (2003) 393–397

Herlocker, J.L., Konstan, J.A., Terveen, L.G. & Riedl, J.T. (2004) Evaluating Collaborative Filtering Recommender Systems. *ACM Transactions on Information Systems*, Vol. 22, No. 1, January 2004, Pages 5–53.

Hoashi,K., Matsumoto, K., & Inoue, N. (2003). Personalization of user profiles for content-based music retrieval based on relevance feedback. *ACM Multimedia*, pp.110–119.

Hofmann, T., & Puzicha, J. (1999). Latent Class Models for Collaborative Filtering. *Proceedings of IJCAI'99*.

Jin, X., Zhou, Y. & Mobasher, B. (2005). A Maximum Entropy Web Recommendation System: Combining Collaborative and Content Features. *KDD'05,* August 21–24, 2005, Chicago, Illinois, USA.

Jung, R., & Heckmann, D. (2006). Ambient Audio Notification with Personalized Music. *UM ECAI'06*.

Karypis, G. (2002). *Cluto A Clustering Toolkit Manual*. University of Minnesota, Department of Computer Science Technical Report.

Kumar, R., Raghavan, P., Rajagopalan, S., & Tomkins, A. (2001) Recommendation Systems: A Probabilistic Analysis. *Journal of Computer and System Sciences*. 63, 42-61 (2001)

Kuo, F.F., Chiang, M.F., Shan, M.K.,& Lee, S.Y. (2005) Emotion-based Music Recommendation by Association Discovery from Film Music. *Proc. 13th annual ACM international conference on Multimedia,* Hilton, Singapore, 507 - 510.

Lekakos, G. & Giaglis, G.M. (2007) A Hybrid Approach for Improving Predictive Accuracy of Collaborative Filtering Algorithms. *User Model User-Adap Inter.* 17:5–40.

Li, T., & Tzanetakis, G. (2003). Factors in automatic musical genre classification of audio signals. *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA).*

Li, Q., Kim, B.M., Guan, D. H., & Oh, D. W. (2004). A Music Recommender Based on Audio Features. *SIGIR'04,* July 25-29, 2004, Sheffield, South Yorkshire, UK.

Li, Q., Myaeng, S.H., Guan, D.H., & Kim, B.M. (2005) A Probabilistic Model for Music Recommendation Considering Audio Features. *Information Retrieval Technology.* Lecture Notes in Computer Science, Volume 3689/2005, pp. 72-83

Li, Q., Myaeng, S. H., & Kim, B. M. (2007). A probabilistic music recommender considering user opinions and audio features. *Information Processing and Management*, 43, 473–487.

Lia, Y., Lu, L., & Xuefeng, L. (2005). A hybrid collaborative filtering method for multiple-interests and multiple-content recommendation in E-Commerce. *Expert Systems with Applications,* 28, 67–77.

Linden, G., Smith, B., & York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing,* 4(1).

Logan, B. (2004). Music recommendation from song sets. *Proc. of the International Conference on Music Information Retrieval (ISMIR) 2004.*

McCarthy, K., Salao, M., Coyle, L., McGinty, L., Smyth, B., & Nixon, P. (2006) Group Recommender Systems: A Critiquing Based Approach. *IUI'06,* January 29–February 1, 2006, Sydney, Australia.

McKay, C., McEnnis, D. & Fujinaga, I. (2006). A Large Publicly Accessible Prototype Audio Database for Music Research. *Proc. of the International Conference on Music Information Retrieval (ISMIR) 2006.*

Pavlov, D.Y., & Pennock, DM. (2002). A Maximum Entropy Approach To Collaborative Filtering in Dynamic, Sparse, High-Dimensional Domains. *Neural Information Processing Systems (NIPS) 2002.*

Pavlov, D. Y., Manavoglu, E., Giles, C. L., & Pennock, D. M. (2004). Collaborative Filtering with Maximum Entropy. *IEEE Intelligent Systems*, Volume: 19, Issue: 6, 40- 47.

Pazzani, M. J. (1999) A Framework for Collaborative, Content-Based and Demographic Filtering. *Artificial Intelligence Review*. 13 (5/6), 393-408.

Popescul, A., Ungar, L. H., Pennock, D. M., & Lawrence, S. (2001). Probabilistic Models for Unified Collaborative and Content-Based Recommendation in Sparse-Data Environments. *Proc. of the Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI-2001).*

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994) GroupLens: An Open Architecture for Collaborative Filtering of Netnews. *Proceedings of the Conference on Computer Supported Cooperative Work*, Chapel Hill, NC, 175-186.

Rolland, P.Y. (2001) Adaptive User Modeling in a Content-Based Music Retrieval System. *Proc. of the International Conference on Music Information Retrieval (ISMIR) 2001.*

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001) Item-Based Collaborative Filtering Recommendation Algorithms WWW10, May 15, 2001, Hong Kong.

Shardanand, U., & Maes, P. (1995). Social information filtering: Algorithms for automating "Word of Mouth". *ACM CHI'95 Conference on Human Factors in Computing Systems*, 210–217.

Smyth, B. & Cotter, P. (2000). A Personalized Television Listings Service. *Communications of the ACM.* August 2000/Vol. 43, No. 8

Smyth, B. & Cotter, P. (2000). A personalised TV listings service for the digital TV age. *Knowledge-Based Systems*. 13, 53-59.

Typke, R., Wiering, F., & Veltkamp, R. C. (2005). A Survey of Music Information Retrieval Systems. *Proc. of the International Conference on Music Information Retrieval (ISMIR) 2005.*

Tzanetakis, G., & Cook, P. (2002). Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302.

Uitdenbogerd, A., & van Schyndel, R. (2002). A review of factors affecting music recommender success. *Proc. of the International Conference on Music Information Retrieval (ISMIR)* 2002.

Vignoli, F., & Pauws, S. (2005). A Music Retrieval System Based on User-Driven Similarity and Its Evaluation. *Proc. of the International Conference on Music Information Retrieval (ISMIR)* 2005.

Wang, J., de Vries, A. P., & Reinders, M. J.T. (2006). A User-Item Relevance Model for Log-Based Collaborative Filtering. *ECIR 2006*, LNCS 3936, pp. 37–48.

Yapriady, B., & Uitdenbogerd, A.L. (2005). Combining Demographic Data with Collaborative Filtering for Automatic Music Recommendation. *Knowledge-Based Intelligent Information and Engineering Systems, LNCS* Volume 3684/2005.

Yoshii, K., Goto, M., Komatani, K., Ogata, T., & Okuno, H.G. (2006). Hybrid Collaborative and Content-based Music Recommendation Using Probabilistic Model with Latent User Preferences. *Proc. of the International Conference on Music Information Retrieval (ISMIR)* 2006.