IMPROVING CONTENT-BASED SIMILARITY MEASURES BY TRAINING A COLLABORATIVE MODEL

Richard Stenzel, Thomas Kamps
Fraunhofer IPSI
Dolivostr. 15
D-64293 Darmstadt
stenzel@ipsi.fraunhofer.de
kamps@ipsi.fraunhofer.de

ABSTRACT

We observed that for multimedia data – especially music - collaborative similarity measures perform much better than similarity measures derived from content-based sound features. Our observation is based on a large scale evaluation with >250,000,000 collaborative data points crawled from the web and >190,000 songs annotated with content-based sound feature sets. A song mentioned in a playlist is regarded as one collaborative data point. In this paper we present a novel approach to bridging the performance gap between collaborative and content-based similarity measures. In the initial training phase a model vector for each song is computed, based on collaborative data. Each vector consists of 200 overlapping unlabelled 'genres' or song clusters. Instead of using explicit numerical voting, we use implicit user profile data as collaborative data source, which is, for example, available as purchase histories in many large scale e-commerce applications. After the training phase, we used support vector machines based on content-based sound features to predict the collaborative model vectors. These predicted model vectors are finally used to compute the similarity between songs. We show that combining collaborative and content-based similarity measures can help to overcome the new item problem in e-commerce applications that offer a collaborative similarity recommender as service to their customers.

Keywords: collaborative metadata, content based sound feature similarity measures, music similarity, machine learning, acoustic measures, evaluation, recommender

1 INTRODUCTION

From the point of view of e-commerce, organizing and recommending items is an important but challenging task. Collaborative metadata constitute an important basis to this end: Web server logs of our customers in the area of newspaper publishing showed that 60 to 70% of the links of a page are followed due to similarity recommendations like "people who liked this, also liked ...".

In today's e-commerce shops for music, two dominant approaches to generating similarity recommendation links can be found: (1) a genre-based approach, where each song is assigned to one or more musical genres and (2) a collaborative similarity approach, where similarity is computed based on purchase histories.

The genre-based approach needs to label each song by hand with some high level genre information like jazz, blues, classical etc. Most shops (e.g. Amazon.com, mp3.com, buy.com, Musicload.de, Musicline.de) provide a genre-based navigation schema. Because a shop may sell more than 800,000 songs, a small number of high level categories would result in huge song classes and are therefore mostly not sufficient. In order to create smaller classes, some shops use more fine grained categories. Musicline.de, for example, uses about 220 genre classes for categorisation. Such a system has some severe drawbacks. First, all songs have to be labelled manually. Second, even with 220 genres and a perfectly balanced distribution of the 800,000 songs across all genres, each genre still would contain an average of 3,636 songs, which is far too large an amount for a user to navigate. As a further drawback it has to be considered that effective access by fine-grained genres is only possible for a musicologist. Finally, genres are highly subjective. As [1] has shown, given 20 genres, only 70% percent of all songs are correctly classified by test users.

Currently much research is being done in assigning genres to songs automatically (e.g. [1,2,3]). Based on our experiments with the genre information of the German Amazon product catalogue, we come to the conclusion that their genre classification is not well suited for similarity search in large music databases. Our evaluation (Section 4) shows, that when tested against real life music listening habits, Amazon-genre based similarity is outperformed by pure content-based sound-feature similarity.

The second area of research is similarity recommendation using collaborative data. Amazon, for example, provides a list of recommendations for each music CD. This item-to-item recommendation is based on data mining in the history of purchases [4]. Even if the results of this collaborative similarity recommender are considered to be good, this approach has the drawback of the new item problem. It is not possible to recom-

mend new music CDs, because shortly after their release they do not occur in enough (or any) purchase histories yet.

In this paper we present a novel approach which attacks the new item problem of collaborative similarity recommendations, i.e. a methodology to compute a list of recommendations of songs given only one song without purchase histories. The methodology described in this paper is a first step towards a CF/CN (collaborative filtering/content-based) meta-level recommender according to Burke [5].

Because of the lack of large purchase histories, we have crawled the web for collaborative data. In the domain of music, collaborative data can be found when searching for personal music playlists, similar to [6] and [7]. Our work is therefore based a large scale evaluation with >250.000.000 collaborative data points, each data point representing one song mentioned in a playlist and >190.000 sound feature sets, each set representing the music of one song. From analysis of the collaborative data points, we created 200 unlabelled and overlapping genres. Our fully automated approach generates as a first step a collaborative model. The principal idea is that this collaborative model consists of one model vector per song. A model vector itself consists of 200 components, each reflecting one of our predefined unlabelled and overlapping genres. Each component of such a model vector contains the binary information, whether a song belongs to a specific unlabelled genre or not. This is decided according to the analysis of the playlists. Each song is assigned to a minimum of 4 and a maximum of 33 genres. Similarity between two songs is defined as the similarity between their model vectors. The major advantage of the generated model, besides its predictive quality, is the ease of prediction given only content-based sound features.

Having both the collaborative model vectors and the content-based sound features for a song, it is possible to train supervised classifiers (support vector machines), which predict for any given sound feature set the 'genres' to which it belongs. Even though we currently use only 30 sound features generated by the MARSYAS framework [1], our experiments show that our method outperforms the Amazon genre-based similarity recommender as well as similarity recommenders based on pure sound feature data. Due to the fact that a supervised machine learning approach is used, all possible sound feature extractors can be used to improve our system. Even hand labelled genre data can be added to create better recommendations.

2 RELATED WORK

The methodology presented in this paper is an attempt to combine two groups of similarity recommenders. One group aims to create similarity measures based on content-based sound features. The other group aims to calculate similarities from collaborative metadata.

Our work heavily depends on the generation of descriptive content-based sound features created from audio signals. First attempts to create similarity measures based on raw audio signal analysis have been undertaken by [8]. The system used a FFT (Fast Fourier Transformation) to compute windowed MFCCs (Mel-Frequency Cepstrum Coefficients) which were quantized and accumulated in histograms. These histograms were used to compute similarities, using the cosine of two histogram vectors. Later on, more complex sound features were used. Welsh et al.'s approach [9], for example, used 1248 sound features per song which, in addition to frequency and amplitude, derived higher level concepts such as temporal data. Another system [10] aims at generating playlists based on content-based features.

Instead of creating similarity measures, some research focuses on automatic genre classification based on raw audio data. Most systems (e.g. [1,2,3]) use a two step approach. First of all, sound features from raw audio data are extracted. In a second step, supervised classification is used to categorize feature vectors into genres. From the viewpoint of similarity search, the classification into genres is an ideal test bed for evaluation of the descriptiveness of sound features.

The use of collaborative metadata is a much investigated area. Most research uses explicit votes on Likert-scales to predict future user voting (e.g. [11]). This might be due to the existence of the MovieLens Dataset, which acts as a perfect test bed for evaluations. In this paper we focus on music similarities, an application area where a gold standard for evaluation does not exist [12]. In [6] several similarity measures are compared. One of their conclusions is, that similarity measures derived from subjective data (co-occurrence analysis of playlists) is the most useful gold standard. Accordingly to this, we use subjective collaborative data as gold standard for our evaluation.

Another interesting collaborative approach, which we do not consider in our work, is the use of textual metadata from the web for artist classification [13,14].

3 METHOD

An item-to-item recommender needs to measure similarity between individual songs. One possibility is to derive a similarity measure from collaborative metadata like purchase histories [4] or playlists. This method produces good results but has the "new item problem". Another option is to compute a similarity measure calculated from content-based features like rhythm and spectral features [9]. Content-based similarity measures do not have the "new item problem" but do not produce as good results as collaborative approaches [6].

In our basic approach we first build a collaborative model from playlists we crawled from the web. In order to produce good results and to avoid the "new item problem" of collaborative recommenders, we then feed content-based sound features (generated by Marsyas [1]) into an auto-learning process that trains our collaborative model. This training is carried out by a machine learning process that maps content-based sound

features to collaborative model vectors. The results are clearly improved content-based similarity measures.

3.1 A Collaborative Item-to-Item Model

Given a set of songs $S = \{ s_1, s_2 \ldots s_K \}$ and a set of playlists or purchase histories $\mathcal{S} = \{ S_i \mid S_i \subseteq S \}$ we compute a collaborative model $\mathcal{C} = \{ c_1, c_2, \ldots c_k \}$ which assigns a collaborative model vector c_i to each song s_i.

Many researchers use supervised classification to predict a musical genre for each song ([1,2,3]). An extension to this approach is to use multiple classifiers which classify each song to multiple overlapping 'genres'. Two songs which share the most 'genres' should be the most similar. This leads to the idea that our collaborative model consists of multiple overlapping 'genres', i.e. a collaborative model vector $C \in \mathcal{C}$ is assigned to each song and each component C_i of C is related to one 'genre'. If a song belongs to a genre, the corresponding component C_i is set to 1, otherwise to 0.

In order to assign a collaborative model vector $c_i \in \mathcal{C}$ to each song $s_i \in S$, we proceed in two steps. In step (a) unlabelled 'genres' are generated by unsupervised clustering of playlists. In step (b), based on the occurrence of songs in playlists which occur in 'genre' clusters, each song is assigned to a set of 'genres'.

The unsupervised clustering algorithm in step (a) computes a projection $\mathcal{S} \rightarrow P_{\mathcal{S}}$ which assigns each playlist $S_i \in \mathcal{S}$ to an unlabelled 'genre' cluster P_j using an unsupervised k-means clustering algorithm [15].

$$P = \mathcal{S} \rightarrow P_{\mathcal{S}} : \{P_j \mid P_j(\mathcal{S}) \subseteq \mathcal{S} \wedge P_j(\mathcal{S}) \cap P_k(\mathcal{S}) = \emptyset\} \quad (1)$$

While clustering, we treat each playlist $S_i \in \mathcal{S}$ as a vector $v \in \mathbf{R}^K$ with each component $v_k = |\{s_k \mid s_k \in S_i\}|$. Similarities and distances are computed according to the Euclidean norm. Because of the sparseness of the playlist data (our vectors contain 99.988% zeros) an iterative approach has been used.

1) Randomly pick one playlist vector $S_i \in \mathcal{S}$ for each playlist cluster P_j as initial centroid
2) Compute the centroid $P_{j,center}$ for each playlist cluster P_j

$$\vec{P}_{j,center} := \frac{1}{\| P_j \|} \sum_{S_i} \frac{S_i}{\| S_i \|} \quad (2)$$

3) Assign each playlist $S_i \in \mathcal{S}$ to the closest centroid P_j using the cosine as similarity function.

$$\cos(\vec{S}_i, \vec{S}_j) = \frac{\vec{S}_i \cdot \vec{S}_j}{\| \vec{S}_i \| * \| \vec{S}_j \|} \quad (3)$$

4) Goto 2) until no more changes occur

In step (b) of the generation of the collaborative item-to-item model, a model vector D is computed for each song $s_s \in S$. Given a set of clusters $P_{\mathcal{S}}$ of playlists containing songs, each component D_k of D consists of the number of playlists in cluster $P_k \in P_{\mathcal{S}}$ containing the song $s_s \in S$.

$$D_k := |\{ S_i \mid s_s \in S_i \wedge S_i \in P_k \}| \text{ with } 1 \le k \le |P_{\mathcal{S}}| \quad (4)$$

Due to the fact that binary classification is more accurate than prediction of real numbers, all nonzero components in D_k could be set to 1, i.e. whenever a song occurs in a playlist cluster P_k, the corresponding vector component D_k could be set to 1, otherwise to 0. This discretisation leads to bad results, because playlists are highly noisy, which means that some songs occur in too many clusters only once or twice. Therefore, we have used a threshold, i.e. we compute a vector $C \in \mathcal{C}$ for each song where all components C_k are set to 1 if $D_k >$ threshold value and set to 0 otherwise.

$$C_k = \begin{cases} 1 & if \quad 0.01 < \dfrac{D_k}{\|D\|} \\ 0 & otherwise \end{cases} \quad with \quad 1 \le k \le \|P_S\| \quad (5)$$

The threshold value of 0.01 has been determined empirically by maximizing in our evaluation results (Section 4). The evaluation will show that each step described above reduces the predictive quality of the collaborative model when using the cosine as similarity measure. The collaborative model nonetheless makes better predictions than a pure sound-feature-based recommender.

3.2 Predicting a Model Vector from Raw Sound Data

In the previous section we generated a collaborative model $\mathcal{C} = \{c_1 \ldots c_k\}$ which assigns each song $s_i \in S$ a collaborative model vector $c_i \in \mathcal{C}$. In this section we use machine learning techniques to learn a mapping from raw sound data of song $s_i \in S$ to a collaborative model vector $c_i \in \mathcal{C}$. Having learned this mapping, it is possible to predict collaborative model vectors for songs, which are not in the initial training set S, i.e. once the mapping is learned it is possible to compute 'collaborative' similarities between songs using the collaborative model even for songs which do not occur in playlists $S_i \in \mathcal{S}$.

There are two steps to learning a mapping from raw sound data of songs $s_i \in S$ to model vectors $c_i \in \mathcal{C}$. First we have to extract content-based sound features $f_i \in \mathcal{F}$ from each song $s_i \in S$. In a second step $|P_{\mathcal{S}}|$ classifiers are trained to map feature vectors $f_i \in \mathcal{F}$ to components of model vectors $c_i \in \mathcal{C}$.

The first step, content-based feature extraction, is currently based on the Marsyas Framework [1], which extracts 30 real valued features from each sound file $s_i \in S$, resulting in a feature vector $f_i \in \mathcal{F}$ for each song. These 30 real valued features, which are called 'genre-features' in Marsysas, include averaged timbral texture features (e.g. spectral centroid), rhythmic content features (e.g. beats per minute) and pitch content features. In principle, other kinds of features can be added, regardless of whether they are real valued or binary (e.g. existing hand labelled classifications).

In the second step we train $|P_S|$ binary support vector machine (SVM) classifiers ($\mathbb{R}^{N=30} \rightarrow [0,1]$), one for each component C_i of $C \in \mathbb{C}$ using the WEKA toolkit [16]. Good parameters for the SVM RBF kernel (support vector machine with radial basis function kernel), are identified by means of a grid search.

After the training phase, similarities between two new songs s'_1 and s'_2 which do not occur in any playlist $S_i \in \mathbb{S}$ can be calculated using the collaborative model. This is done in three steps.

1. ($S \rightarrow \mathbb{F}$) Extraction of the sound features $f'_1, f'_2 \in \mathbb{F}'$ from songs s'_1 and s'_2, based on the audio-signal
2. ($F \rightarrow C_i$ with $F \in \mathbb{F}' \wedge C \in \mathbb{C}'$) Prediction of the two model vectors $c'_1, c'_2 \in \mathbb{C}'$ by applying the $|P_S|$ trained SVMs to the sound features vectors f'_1 and f'_2, i.e. each component of $c'_{1,i}$ and $c'_{2,i}$ of c'_1 and c'_2 is predicted independently.
3. Computation of the similarity between c'_1 and c'_2, which is defined as the cosine of c'_1 and c'_2.

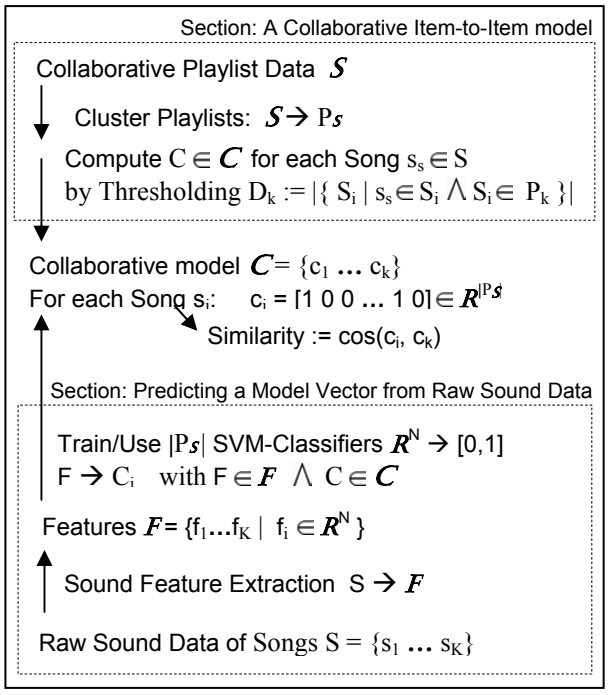

Figure 1: Overall Block Diagram of our Method

By predicting the collaborative model vectors from sound features, collaborative recommenders can be seamlessly extended with content-based similarity to deal with the new item problem. Our method can be employed for completely new songs. For songs which have a large purchase history, a purely collaborative approach can be used. For songs which have not been sold very often, a combination of collaborative and content-based methods could be used. The combination could be arrived at, for example via a weighted sum of two model vectors $C \in \mathbb{C}$, where one vector is based on our method, the other one on collaborative data.

The use of binary components in $\mathbb{C}$ and the independent training of SVMs for each component C_i of $C \in \mathbb{C}$ is not optimal. This is due to the fact that the components D_i (4) of our collaborative model vectors D are not mutually independent, because each component of a vector D is based on a cluster of playlists. Playlist clusters themselves have similarities to each other, i.e. it is likely that if a song occurs in one playlist cluster it will occur in the most similar clusters too. This means that if one component D_i (=cluster) in D is set to a specific value, the most similar components of D will have similar values. It would have been nice to train a mapping from sound features $F \in \mathbb{F}$ to D without any discretizations ($D \rightarrow C$) using an algorithm, which learns vector-valued functions ($\mathbb{R}^N \rightarrow \mathbb{R}^{|P_S|}$). Unfortunately, research on learning vector-valued functions is still at a rather early stage [17]. A simpler approach is to neglect the mutual dependence of components D_i of D and train independent classifiers to predict binary classes of C_i.

4 EXPERIMENTS

In our experiments we build and evaluate four types of similarity measures, varying in their data sources. These similarity measures are based on data from

- Collaborative sources
- Content-based sound features
- Amazon's genre data
- Collaborative sources and content-based sound features (our method)

Based on the similarity measures a song recommender for each measure is created. Each recommender computes for each given song $s_i \in S$ the N most similar songs $TopN = \{t_1 \ldots t_N \mid t_i \in S\}$. In order to evaluate the predictive performance of each recommender, we follow the observations of Berenzweig [6] that collaborative data sources are the best gold standard. This means that our evaluation metric is based on collaborative playlist data $\mathbb{S}_{test} = \{ S_{i,test} \mid S_{i,test} \subseteq S\}$ which is not used in the training phase. Precision, Recall and F1 are calculated for each song $s_{song} \in S$ and averaged over all songs.

$$precision = \frac{1}{\left|\{S_{i,test} \mid s_{song} \in S_{i,test}\}\right|} \sum_{S_{test}} \frac{\left|S_{i,test} \cap TopN\right|}{\left|TopN\right|} \quad (6)$$

$$recall = \frac{1}{\left|\{S_{i,test} \mid s_{song} \in S_{i,test}\}\right|} \sum_{S_{test}} \frac{\left|S_{i,test} \cap TopN\right|}{\left|S_{i,test}\right|} \quad (7)$$

$$F1 = 2 * \frac{precision * recall}{precision + recall} \quad (8)$$

F1 is the classic information retrieval measure for overall retrieval performance. In our evaluation metrics "precision" reflects the probability given one song by the user that this user would add the recommended song in a playlist together with the given one.

In the following subsection we show that it is possible to create collaborative similarity measures by crawling music playlists from the web. We show in a large scale evaluation that the quantity of the crawled data if a key factor regarding the predictive performance of this approach. In Section 4.2 we create and evaluate a similarity measure like proposed in Section 3.1, which is based on crawled collaborative data. As a testing baseline we create and evaluate similarity measures based on pure content-based sound features and based on Amazon's genre-data in Section 4.3. Finally in Section 4.4 we train and predict a collaborative model based on content-based sound features like proposed in section 3.2.

4.1 Crawling the Web for Music Playlists

In this section a song item-to-item recommender based on 250,000,000 collaborative data points crawled from the web is built and evaluated. We show that 'size matters', i.e. the quality of recommendation increases with the quantity of collaborative data or, more formally, that the averaged F1 measure of the overall system is proportional to log(|Data|).

A collaborative model of music similarity which can be tested against real world data (e.g. user playlists, radio programme histories) should cover a large portion of current music. [7], for example, used a 2800 album dataset which was used by a closed community. Because of the lack of a free dataset, we choose to crawl the web for real world playlists.

Some programs (e.g. WinAmp, XMMS) produce HTML pages with playlists. Due to the low cost of publishing web pages, many people put "the playlist of our last party" online. These pages have simple structures and can be found using common search engines (e.g. altavista, google). The idea is to search for specific words within the auto generated HTML page (e.g. for WinAmp +"WinAmp generated playlist") in combination with artist names (e.g. +"Eric Clapton"). Artist names can be taken from a music catalogue, such as Amazon. Unfortunately, not all playlists contain valuable information, e.g. some contain "all tracks I have on my harddisk". To decide, if a playlist should be used, we employed a rule based filter. The rules are: a playlist must contain at least three different artists, have at most 300 entries and tracks may not be sorted alphabetically.

Due to limitations of search engines, such as a restriction on queries per day and computer, the crawler has to run in a distributed manner and should automatically crawl the immediate area of a match. After two years of crawling we have gathered >250,000,000 playlist member datasets.

The listed songs in playlists do not have globally identifiable keys. Usually a playlist entry contains only a string like "track01.mp3" or "(eagles)-the long run-04 the disco strangler.ogg". Our first attempt to match the text strings (>17.000.000) with the free CDDB failed, because the CDDB is not itself normalized (e.g. "Track01" is a common track name in the CDDB). In order to produce high quality results, we have used the Amazon product catalogue as a dictionary to match against. Even if Amazon sells only a small portion of existing music, and not all CDs in the catalogue contain track information, we were able to match about 7.5% of our playlist entries against this reliable source (compared to 39% using the CDDB, where "Track01" is included).

The result of crawling and matching is a data set with 176,930 songs, 880,985 playlists and 18,769,469 playlist entries. Compared to other research systems, our approach (>8,000,000 playlists before tagging and filtering) is an extremely large scale approach. E.g. [6], who stated that the similarity measures created with playlist data, is the "most useful ground truth" (gold standard), used only 29,000 playlists from "The Art of the Mix" (www.artofthemix.org), which is a small subset of our data (about 0.3% of our data set).

To evaluate the improvement in prediction accuracy with the increase of data, we have produced a simple artist similarity recommender. The similarity between two artists is defined by the cosine between their playlist occurrence vectors. Test data was taken to compute an averaged F1 measure. The evaluation results are shown in Figure 2. The F1 measure is approximately proportional to log(|Data|).

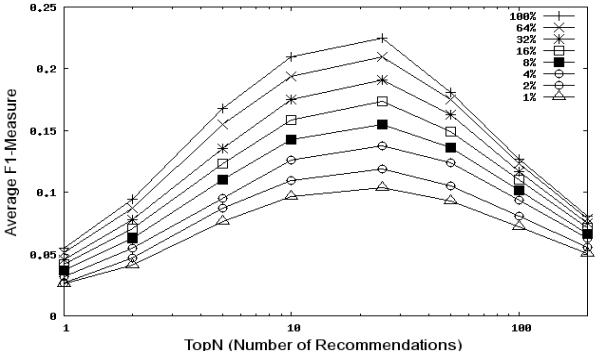

Figure 2: Each line represents a collaborative artist similarity recommender, evaluated against playlists. The lowest line is built from 1% of all available playlist data, the highest from 100%. The F1 measure (Y-Axis) increases roughly with log(|Data|). The number of recommended Items (X-Axis) influences F1.

Due to the fact, that we did not know whether the results are specific to the cosine algorithm, we used an association rules algorithm and some variants. The different algorithms did not change the result in principle. The normalisation rules, such as the rule that every artist occurs only once in a playlist, have more influence on the results than alternative algorithms have.

In addition to the automated evaluation against crawled playlists, we ran a web based test, where three different recommender engines were randomly presented to test users. The test users had to express their satisfaction with the recommended results. The three recommender engines only differ in the amount of data used to train them (1%, 8% and 100% of all playlist

data). Interestingly, even if the test users did not know which engine was presented, their ranking corresponded with the F1 measures of our automated evaluation.

4.2 Creating a Collaborative Model

With 30 content-based features per song provided by the Marsyas framework we cannot realistically train high dimensional collaborative models. Thus we need to reduce the dimension of the collaborative model. To investigate the effect of reducing dimensions we have performed the following experiment:

Using the playlist data set described above, it is possible to build a collaborative model described in Section 3.1. In the creation process, there are four steps in which a similarity measure between songs can be computed and evaluated against real life playlist data. In each step, a feature vector is assigned to each song. Similarity between two songs can be evaluated using the cosine measure between the feature vectors.

In the first step we used the raw playlist data to create a similarity measure. Even if the cosine measure is not the first choice for computing similarity, we use it to be compatible with the other steps. Each component of a song's feature vector in this step is a playlist. If the song occurs in the playlist, the component is set to 1, otherwise to 0. The evaluation of the resulting similarity measure is the highest line (best performance) in figure 3, marked as 'collaborative'.

The second line from the top, marked as 'clustered songs', shows the performance of the second step, in which the feature vector D is used to create a similarity measure, i.e. at this point all playlists are clustered in $|P_S| = 200$ clusters using the k-means algorithm described in section 3.1. Each vector D for a song $s_{song} \in S$ has 200 components D_i and each of these components contains the number of occurrences of song s_{song} in playlist cluster P_i.

Our experiments show that if we create less than 200 clusters, the predictive precision decreases. Creating more than 200 clusters does not increase the precision. The difference in the results of precision when creating 200 or 400 playlist clusters is about 4±0.5 percent.

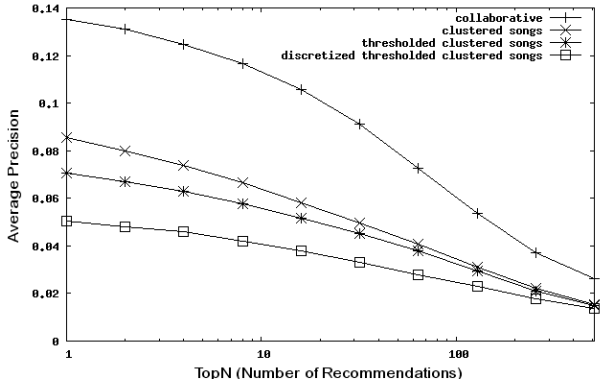

Figure 3: Evaluation of reduction steps described in Section 3.1 against a test set of playlists.

The third step and third line in figure 3 uses D/||D|| as feature vector with all components <0.01 set to 0. And finally, the bottom line in figure 3 uses the binary feature vector $C \in \mathbb{C}$ as source for the similarity measure, where the feature vector from step 3 is taken and all components >0 are set to 1.

It is obvious that each step decreases the predictive performance of the similarity measure. On the other hand, even the bottom line of the collaborative model performs far better than purely content-based similarity measures (Section 4.3).

4.3 Comparing a Content-based and a Genre-based Music Recommender

In this section we set up a testing baseline by generating and evaluating three similarity measures based on three different data sources. One similarity measure is based on 30 dimensional content-based sound feature data provided by Marsyas [1], the second one is based on Amazon's genre-data and the third is based on a 30 dimensional random vector.

In order to build a content-based music recommender, it is necessary to have a large database of sound features. Due to the lack of a common set of free test songs, or a free large scale collection of sound features, we created a sound feature extraction utility based on Marsyas [1] which extracts a 30 dimensional feature vector per song. The extraction utility, which sends extracted metadata via http to a central server, runs on our students' private music collections. Unfortunately, the collection is not completely balanced across genres, which means that nearly 70% percent of all data is related to the genres Pop / Rock / Independent.

In total we were able to extract >190,000 files. After eliminating duplicates and matching against the Amazon (Germany) product catalogue, we obtained 41,816 tagged feature sets.

For the purpose of performance comparison, we build a simple sound-feature-based music recommender. Similarity between two songs or two feature vectors is defined as the cosine of the two feature vectors. Before the cosine calculation, each feature was linearly scaled to the range [-1...1]. The underlying assumption of this straightforward implementation is that all features contribute equally to the subjective perception of music similarity and all features are equally noisy across all types of music. This assumption makes it hard to add new features from different frameworks and even if the current features are carefully selected the assumption may not hold.

In order to compare the basic sound feature recommender to an alternative approach we build a music similarity recommender based on hand labelled genre information. Because we matched all music titles against the product catalogue of Amazon (Germany), it seems reasonable to use their genre information. The highest level of genre information at Amazon.de divides all music into twelve top level genres, but most songs belong to more than one genre, e.g. "The Beatles – All

You Need is Love" belongs to the genres "Jazz&Blues", "Pop", "Rock" and "Classics". Of course, this is not a musicological categorisation but a real life example.

To create a similarity recommender based on genre information, we represent the genre information as binary feature information, i.e. each genre is one dimension in a song's feature vector. If the song belongs to the genre, that component is set to 1, otherwise to 0. Just as for the sound-feature-based music similarity recommender, similarity between two songs or two feature vectors is defined as the cosine of the feature vectors.

As a testing baseline we employed a similarity engine, with 30 real valued random features per song, with random numbers in the range [0…1]. Again, similarity between two songs or two random-feature vectors is defined as the cosine of the two feature vectors. Even if in theory the predictive precision of random-features should be constant in an equally distributed test set, the real world test bed shows a small increase in precision with an increase in the number of recommended items.

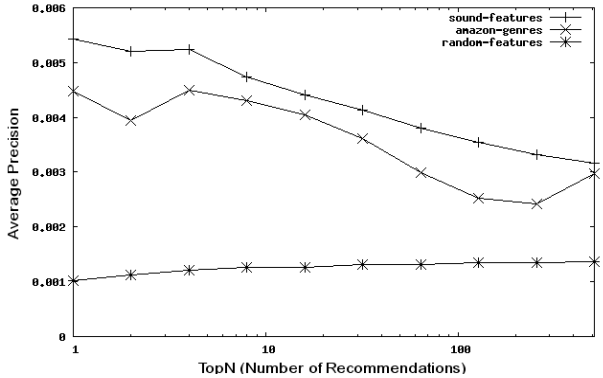

Figure 4: Comparison of three music similarity recommendation engines based on sound features, Amazon (Germany) genres seen as binary features and random features.

All three similarity engines were tested against 654,506 human generated playlists. The results (precision against number of recommendations) are shown in Figure 4. It can be seen, that the sound-feature-based similarity engine as well as the Amazon genre-based engine produce far better results than the random-feature based engine. More interesting is that the sound-feature-based similarity recommender outperforms the hand labelled Amazon genre-based recommender.

4.4 Predicting the Collaborative Model

In this section we create a similarity measure with our method described in Section 3.2. In an evaluation we compare our approach with a content-based approach.

In the previous sections we created a collaborative model which represents each song as a binary feature vector $C \in \boldsymbol{C}$, in our case with 200 binary dimensions. Additionally, we have used the MARSYAS framework to create for each song a content-based feature vector with 30 dimensions. On this basis, we can use a supervised learning approach to learn a mapping from features $\boldsymbol{F}$ to the collaborative model $\boldsymbol{C}$. After the one time training, for each new song a collaborative feature vector $C \in \boldsymbol{C}'$ can be predicted using the learned mapping from $\boldsymbol{F}$ to $\boldsymbol{C}$.

We have collaborative data for 176,930 songs and 41,816 tagged content-based feature sets. The intersection of both sets contains 36,916 songs.

As stated in Section 3, even if the dimensions of each collaborative model vector $C \in \boldsymbol{C}$ are not independent, we have to assume that they are independent because of the lack of appropriate vector-valued learning methods. This means that we trained 200 binary support vector machines using a radial basis function kernel (SVM with RBF kernel) to predict each binary component C_i of $C \in \boldsymbol{C}$ independently, based on the feature vector $F \in \boldsymbol{F}$ provided by MARSYAS.

To evaluate our method we predicted, based on the sound feature vectors F, for each song of our test set data a collaborative model vector $C \in \boldsymbol{C}'$. After this prediction similarities of the predicted vectors C are computed using the cosine as similarity function. The resulting similarities are tested against 301,227 human generated playlists.

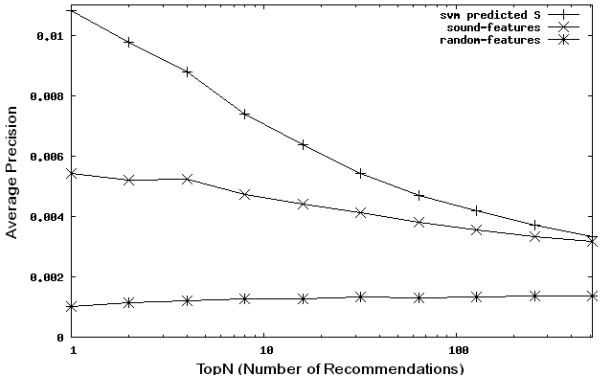

Figure 5: Comparison of three music similarity engines: our method (top line), sound features (middle) and random features (bottom line).

Of course, it would have been preferable to present a ten-fold cross validation instead of only one run, but the training of one SVM takes (incl. automated optimization of the RBF-kernel parameters) about 3.4 hours which results in 28 days for 200 SVMs.

The evaluation results in figure 5 show that our method (top-line) nearly doubles the precision for the 'most similar' item (TopN=1), i.e. given one song by the user, the probability that this user would add the recommended song in a playlist with the given one is doubled by our method compared to a pure content-based approach and more than doubled compared to a recommender based on Amazon's genre-data (Section 4.3).

5 CONCLUSIONS AND FUTURE WORK

In this paper we have presented an approach to improving purely content-based similarity measures by training a collaborative model with content-based sound features. With our approach we were able to almost double the precision of similarity-based recommendations com-

pared to a content-based approach when tested in a real-life test bed against human generated playlists. In addition to that, our approach is open to any progress in the area of content-based feature extraction. The more descriptive new sound features are, the better the result should be.

On the collaborative side, we created a collaborative recommender from a large scale dataset (18,769,469 tagged data points). This recommender and two versions, trained with fewer data, have been tested by users on the web. The users ranked their satisfaction with the recommenders according to the offline predicted F1 measure and precision of the recommender, which means that the test bed seems to reflect reality.

We conducted an experiment to build a similarity measure based on the genre data of Amazon (Germany). The evaluation results of the Amazon genre-based similarity engine were outperformed by the similarities gained through the pure sound-feature-based approach. It emerges that an approach based on Amazon's genres is not suitable.

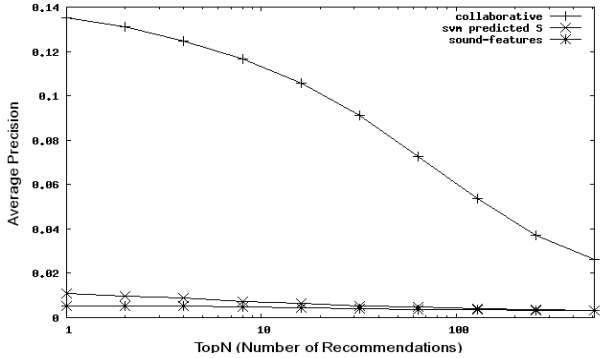

Figure 6: Performance comparison of 3 music similarity recommenders: pure collaborative approach (top line), our approach (middle), sound-feature-based (bottom line).

Fig. 6 shows that our method has much potential for improvement. The theoretical maximum is the top line in figure 6. One way to improve the performance of our method could be to use more descriptive sound features, another one is to build a better collaborative model. A third way to improve the prediction rate is to find machine learning approaches which exploit the interdependence of collaborative model vector components.

Simulations showed that if we could raise our current average SVM prediction rate from 73.1% correctly predicted items to 95%, we could raise the precision for the 'most similar item' (one recommendation) in our method from 0.0108 to 0.04, compared to 0.0054 with a purely sound feature-based approach.

REFERENCES

[1] George Tzanetakis and Perry Cook: "Musical Genre Classification of Audio Signals" IEEE Transactions on Speech and Audio Processing, 10(5), 2002

[2] D. Pye, "Content-Based Methods for Managing Electronic Music". In Proc ICASSP 2000, 2000

[3] T. Lambrou, P. Kudumakis, M. Sandler, R. Speller and A. Linney, "Classification of Audio Signals using Statistical Features on Time and Wavelet Transform Domains". In Proc ICASSP 1998, 1998

[4] G. Linden, B. Smith, "Amazon.com Recommendations: Item-to-Item Collaborative Filtering", in IEEE Internet Computing, pp. 76-80, Jeremy York Publication, January 2003

[5] R. Burke, "Hybrid Recommender Systems: Survey and Experiment", User Modeling and User-Adapted Interaction 12(4): 331-370; Nov 2002

[6] A. Berenzweig, B. Logan, D. Ellis, B. Whitman, "A large-scale evaluation of acoustic and subjective music similarity measures", Proc. ISMIR 2003, 2003

[7] W. W. Cohen, W. Fan, "Web-collaborative filtering: recommending music by crawling the web". WWW9 / Computer Networks, 33(1-6):685–698, 2000.

[8] J. T. Foote. "Content-based retrieval of music and audio."In C.-C. J. Kuo et al., editor, Multimedia Storage and Archiving Systems II, Proc. of SPIE, Vol. 3229, pp. 138-147, 1997

[9] Matt Welsh, Nikita Borisov, Jason Hill, Rob von Behren, and Alec Woo: "Querying Large Collections of Music for Similarity", UC Berkeley Technical Report UCB/CSD-00-1096, 1999

[10] B. Logan, "Content-based playlist generation: Exploratory experiments", Proc. ISMIR 2002, pp. 295-296 (2002)

[11] N. Good, J. Schafer, J. Konstan, A. Borchers, B. Sarwar, J. Herlocker and J. Riedl, "Combining collaborative Filtering with personal agents for better recommendations" In Proceedings of the Sixteenth National Conference on Artificial Intelligence, 1999

[12] D. P. Ellis, B. Whitman, A. Berenzweig and S, Lawrence, "The quest for ground truth in musical artist similarity", Proc. ISMIR 2002, 2002

[13] P. Knees, E. Pampalk, and G. Widmer, "Artist Classification with Web-based Data", Proc. ISMIR 2004, pp 517-524, (2004)

[14] B. Whitman, S. Lawrence "Inferring descriptions and similarity for music from community metadata", in Proc. ICMC 2002, 2002

[15] R. Duda, P. Hart, and D. Storck. "Pattern classification" John Wiley & Sons, New York, 2000.

[16] I. H. Witten, E. Frank: "Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations", Morgan Kaufmann, 1999

[17] C. A. Micchelli, M. Pontil, "On Learning Vector-Valued Functions" in Neural Computation, MIT Press, vol. 17, no. 1, pp. 177-204(28), 2005