A Music Search Engine Built upon Audio-based and Web-based Similarity Measures

Peter Knees[1], Tim Pohle[1], Markus Schedl[1], and Gerhard Widmer[1,2]
[1]Department of Computational Perception, Johannes Kepler University Linz, Austria
[2]Austrian Research Institute for Artificial Intelligence (OFAI)
peter.knees@jku.at, tim.pohle@jku.at, markus.schedl@jku.at, gerhard.widmer@jku.at

ABSTRACT

An approach is presented to automatically build a search engine for large-scale music collections that can be queried through natural language. While existing approaches depend on explicit manual annotations and meta-data assigned to the individual audio pieces, we automatically derive descriptions by making use of methods from Web Retrieval and Music Information Retrieval. Based on the ID3 tags of a collection of mp3 files, we retrieve relevant Web pages via Google queries and use the contents of these pages to characterize the music pieces and represent them by *term vectors*. By incorporating complementary information about acoustic similarity we are able to both reduce the dimensionality of the vector space and improve the performance of retrieval, i.e. the quality of the results. Furthermore, the usage of audio similarity allows us to also characterize audio pieces when there is no associated information found on the Web.

Categories and Subject Descriptors:
H.3.3 [**Information Systems**]: Information Storage and Retrieval; H.5.5 [**Information Systems**]: Information Interfaces and Presentation – *Sound and Music Computing*

General Terms: Algorithms

Keywords: Music search engine, Music Information Retrieval, music similarity, context-based retrieval, cross-media retrieval

1. INTRODUCTION

Over the past years, using text-based search engines has become the "natural" way to find and access all types of multimedia content. While there exist approaches to automatically derive and assign semantic, natural language descriptors for images, videos, and – of course – text, the broad field of (popular) music has not drawn that much of attention. Basically all existing music search systems, e.g. those offered by commercial music resellers, make use of manually assigned subjective meta-information like genre or style (in addition to objective meta-data like artist, album name, track name, or year) to index the underlying music collection. Thus, the person issuing the query (i.e. the potential customer) must already have a very precise conception of the expected result set. The intrinsic problem of these systems is the limitation to a rather small set of meta-data, whereas the musical, or more general, the cultural context of music pieces is not captured.

However, as digital catalogs rapidly become larger and more inconvenient and inefficient to access, the need for more sophisticated methods that enable intuitive searching inside large music collections increases. For example, instead of just finding tracks that are labeled as *rock*, it would be valuable to be able to formulate a query like *"rock with great riffs"* to emphasize the importance of energetic guitar phrases. Another query could be *"relaxing music"* that should return relaxing pieces regardless of which "genre" they are assigned to. Clearly, music resellers with very large music databases or music information systems could benefit from such an engine, as it provides access to their catalog in the most common and most accepted manner.

Recent developments in music information systems respond to the fact that new and possibly unconventional approaches are necessary to support the user in finding desired music. Most of them make use of content-based analysis of the audio files (e.g. [12, 6]) or use collaborative recommendations to point the users to music they might like (e.g. [9], Last.fm[1], or on-line retailers like Amazon). Also, there are techniques that incorporate information from different sources to build interactive interfaces [15, 21].

In this paper, we present first steps toward the challenging task of automatically building a search system that is capable of finding music that satisfies arbitrary natural language queries. For each track in a collection of mp3 files, we retrieve a set of relevant Web pages via Google. This enables us to represent music pieces in a traditional term vector space. Additionally, we make use of a state-of-the-art audio similarity measure. Thus, we combine information about the *context* of music with information about the *content*. With the integration of acoustic similarity, we are able to *(a)* reduce the dimensionality of the feature space, *(b)* describe music pieces with no (or little) related information present on the Web, and *(c)* improve the quality of the retrieval results. The evaluation carried out on a collection consisting of more than 12,000 music pieces gives strong indication that a natural language search engine for music collections is in reach.

SIGIR'07, July 23–27, 2007, Amsterdam, The Netherlands.

[1]`http://www.last.fm`

2. RELATED WORK

Several methods to retrieve music from large databases have been proposed. Most of these approaches are based on *query-by-example* methods (e.g. [17]). Thus, the query must consist of a piece of information that has the same representation as the records in the database. For example, in *Query-by-Humming/Singing* (QBHS) systems [11] the user has to sing or hum a part of the searched piece into a microphone. In most cases, these systems operate on a symbolic representation of music (i.e. MIDI). Other query-by-example systems use a small, low quality recorded portion of a music piece as input, identify the piece, and return the associated meta-data (artist, title, etc.). Typically, such services are accessible via cellular phone (e.g. from the commercial service Shazam[2]).

An even more challenging task is to design systems that enable cross-media retrieval. In our case, systems that allow queries consisting of arbitrary natural language text, e.g. descriptions of sound, mood, or cultural events, and return music pieces that are semantically related to this query, are of interest. Unfortunately, the number of systems enabling its users to perform such queries is very little. The most elaborate approach so far has been presented by Baumann et al. [5]. Their system is supported by a semantic ontology which integrates information about artist, genre, year, lyrics, and automatically extracted acoustic properties like loudness, tempo, and instrumentation and defines relations between these concepts. Beside the correct extraction of the features, also the mapping of the query to the concepts in the ontology has to be accomplished. In the end, the system allows for semantic queries like *"something fast from..."* or *"something new from..."*. Also phonetic misspellings are corrected automatically.

In [8], Celma et al. present the music search engine *Search Sounds*[3]. The system uses a special crawler that focuses on a set of manually defined "audio blogs", which can be accessed via RSS links. In these blogs, the authors explain and describe music pieces and make them available for download (whether legally or illegally depends on the blog). Hence, the available textual information that refers to the music, together with the meta-data of the files, can be used to match text queries to actual music pieces. Furthermore, for all returned pieces, acoustically similar pieces can be discovered by means of content-based audio similarity.

Another system that opts to enhance music search with additional semantic information is Squiggle [7]. In this framework, queries are matched against meta-data and also further evaluated by a *word sense disambiguation* component that proposes related queries. For example, a query for *"rhcp"* results in zero hits, but suggests to search for the band "Red Hot Chili Peppers"; searching for *"Rock DJ"* proposes the song by "Robbie Williams", the genre "Rock", as well as other artists (all "DJs"). The underlying semantic relations are taken from the freely available community databases MusicMoz[4] and MusicBrainz[5]. However, although semantic relations are integrated, the system depends on explicit knowledge which is in fact a more extensive set of manually annotated meta-data.

A system that is not restricted to a pre-defined set of meta-data is Last.fm. Last.fm integrates into music player software and keeps track of each user's listening habits. Based on the collected data, similar artists or tracks can be recommended. Additionally, users can assign tags to the tracks in their collection. These tags provide a valuable source of information on how people perceive and describe music. A drawback of the system is that most tags are highly inconsistent and noisy. We discuss this issue in more detail in Section 4 where we use part of the Last.fm data to evaluate our own approach.

Beside music information systems that deal solely with popular music, there exist a number of search engines that use specialized (focused) crawlers to find all types of sounds on the Web. The traced audio files are indexed using contextual information extracted from the text surrounding the links to the files. Examples of such search engines are Aroooga [16] and FindSounds[6].

Finally, we briefly review approaches that aim to bridge the *semantic gap* for music by extracting information from the pure audio signal and assigning some sort of meta-data. In [15], music landscapes created from audio similarity information are labeled with music related terms extracted from the Web. [21] uses similar techniques to arrange a music collection around a circle for easy access. In [24], low-level characteristics of audio signals are mapped to semantic concepts to learn the "meaning" of certain acoustic properties.

In contrast, in this paper, we directly derive culturally associated descriptors for audio files by extracting the desired information from related Web pages. Additionally, we make use of audio-based similarity to improve this technique. As a result, we obtain a representation of each music piece in a term vector space. To search for music pieces, queries are also transformed to this vector space where distances to all music pieces can be calculated.

3. TECHNICAL REALIZATION

In this section, we describe our technique to build a natural language search engine for music. After a preprocessing step, related Web pages for each track are retrieved. Second, we compute acoustic similarity directly from the audio files which we use in the following steps to reduce the dimensionality of the used vector space and modify the vector representations toward acoustically similar pieces. Finally, we describe how queries are processed in order to find the most adequate pieces in the collection.

3.1 Preprocessing the Collection

To obtain descriptors for the tracks in an mp3 collection, we make use of the information found in the ID3 tags of the files. More precisely, we extract the values of the fields "artist", "album", and "title". Very commonly, additional information is included in these tags, e.g. tracks that are a collaboration of two artists often contain the second artist in the title (indicated by *feat.*, *and*, *with*, etc.) or both artists are mentioned in the artist field. Also other meta-information (e.g. to indicate live versions or remixes) can be found. To avoid too constrained queries in the next step, this extra information is removed. One drawback of this preprocessing step is that it affects also artists like "Ike & Tina Turner", who are afterward represented only by "Ike".

[2] http://www.shazam.com
[3] http://www.searchsounds.net
[4] http://www.musicmoz.org
[5] http://www.musicbrainz.org
[6] http://www.findsounds.com

	artist	*album*	*title*
mean	5,042,732	39,891	15,844
lower quartile	50,900	0	14
median	386,000	12	167
upper quartile	1,260,000	461	845
total number	1,200	2,073	12,601
$\lvert count = 0 \rvert$	8 (0.7%)	860 (41%)	2,430 (19%)
$\lvert count \leq 100 \rvert$	22 (1.8%)	1,360 (65%)	5,533 (44%)

Table 1: Statistics of estimated page counts from Google for our evaluation collection.

Based on the meta-tags, also pieces that are likely to contain only speech are ignored (e.g. in Rap music this is often indicated by the word *Skit*) as well as all tracks named *Intro* or *Outro* and tracks with a duration below 1 minute. Furthermore, all duplicates of tracks are excluded from the next steps to avoid unnecessary similarity computations and redundancies (different versions of tracks could be displayed, for example, as alternatives in the retrieval results). Among all duplicates, the version containing no meta-information like *live* or *remix* is chosen for further processing. In future work, we will also elaborate methods to deal with remixes, i.e. the fact that for remixes in general the remixing artist is more important than the original artist.

3.2 Web-based Features

The primary source of information for our approach is the Web. Previous work that uses Web data for Music Information Retrieval [25, 14] operates only on the artist level. The main argument for this limitation is the claim that there is not enough specific information present on the Web for each individual track. Table 1 shows some statistics on the estimated numbers of available Web pages returned by Google. It can be seen that the number of available pages decreases drastically when searching for album or title instead of artist alone. (Please note that the collection contains many tracks from samplers and compilations, which explains the low page counts for album related queries.) For our task, the number of queries that return zero or below 100 pages are of particular interest. We can see that very low page counts occur only sporadically when searching for artist related pages (around 1%). On the track level for 19% no information is found at all. Thus, to gather as much track specific information as possible while preserving a high number of available web pages, we decided to combine the results of three queries issued to Google for each track in the collection:

1. "*artist*" music
2. "*artist*" "*album*" music review
3. "*artist*" "*title*" music review -lyrics

The first query is intended to provide a stable basis of related documents. With the second query, we try to find reviews of the album on which the track was published, cf. [25]. The third query targets very specific pages and aims at preventing lyrics to be included. We plan to address song lyrics separately in future work. For each query, at most 100 of the top-ranked Web pages are retrieved and joined into a single set.[7] All retrieved pages are cleaned from HTML tags and stop words in six languages.[8]

For each music piece m and each term t appearing in the retrieved pages, we count the number of occurrences tf_{tm} (term frequency) of term t in documents related to m. Additionally, we count df_{tm} the number of pages related to m the term occurred in (document frequency). All terms with $df_{tm} \leq 2$ are removed from m's term set. Finally, we count mpf_t the number of music pieces that contain term t in their set (music piece frequency). For pragmatic reasons, we further remove all terms that co-occur with less than 0.1% of all music pieces, resulting in a vector space having about 78,000 dimensions. To calculate the weight $w(t,m)$ of a term t for music piece m, we use a straight forward modification of the well-established term frequency $\times$ inverse document frequency ($tf \times idf$) function [23]:

$$w(t,m) = \begin{cases} (1 + \log_2 tf_{tm}) \log_2 \frac{N}{mpf_t} & \text{if } tf_{tm} > 0 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where N is the number of music pieces in the collection. As can be seen, we treat all Web pages related to a music piece as one large document. The resulting term weight vectors are normalized such that the length of the vector equals 1 (Cosine normalization). This removes the influence of the length of the retrieved Web pages as well as the different numbers of retrieved pages per track.

3.3 Audio-based Similarity

To complement the extracted Web-based features with perceived acoustical similarity of the associated music, we follow a well-established procedure, e.g. [20]: For each audio track, *Mel Frequency Cepstral Coefficients* (MFCCs) are computed on short-time audio segments (called *frames*) to describe the spectral envelope of each frame and model thus *timbral* properties. We use the definition given in [2]:

$$c_n = \frac{1}{2\pi} \times \int_{\omega=-\pi}^{\omega=+\pi} \log\left(S\left(e^{j\omega}\right)\right) \cdot e^{j\omega \cdot n} d\omega \quad (2)$$

As proposed, we calculate 19 MFCCs on each frame. Ignoring the temporal order of frames, each song is then going to be represented as a Gaussian Mixture Model (GMM) of the distribution of MFCCs [3]. According to Mandel and Ellis [18], a Single Gaussian Model with full covariance matrix is sufficient for representation, which facilitates computation and comparison of the models. Instead of estimating similarity of GMMs via Monte Carlo sampling, a symmetrised *Kullback-Leibler divergence* can be calculated on the means and covariance matrices [18].

However, as described in [1, 22], the Kullback-Leibler divergence has some undesirable properties. For example, it can be observed that some particular pieces, so called hubs, are frequently "similar" (i.e. have a small distance) to many other pieces in the collection without sounding similar. On the other side, some pieces are never similar to others. Furthermore, the Kullback-Leibler divergence does not fulfill the triangle inequality.

[7] For evaluation, pages from Last.fm are kept out of this set, as we use information from Last.fm/Audioscrobbler later to measure the quality of our approach.

[8] English, German, Spanish, French, Italian, and Portuguese

To cope with these issues imposed by a distance measure that is no metric, we developed a simple rank-based correction called *Proximity Verification* [22]. As a consequence, all further steps presented here will be based on the ranking information of the audio similarity measure only, i.e. it will only be of interest whether a piece is most similar, second similar, third similar, etc. to a piece under consideration and not to which numerical extent the two pieces are considered to be similar.

3.4 Dimensionality Reduction

To reduce the number of terms and remove irrelevant dimensions from the term vector model, we use the χ^2 test which is a standard term selection approach in text classification, e.g. [26]. Usually, the χ^2 test needs some information on class assignments to measure the independence of a term t from a class c. In our case, no class information (e.g. genre) is available. Hence, we make use of the derived audio similarity. For each track, we define a 2-class term selection problem and use the χ^2 test to find those terms that discriminate s, the group of the most similar tracks (which we assume to be homogeneous according to the audio similarity measure), from d, the group of the most dissimilar tracks (which is not necessarily homogeneous). We define s and d to comprise 100 tracks each, to have a solid number of documents for the χ^2 test. For each track, we calculate

$$\chi^2(t,s) = \frac{N(AD-BC)^2}{(A+B)(A+C)(B+D)(C+D)} \tag{3}$$

where A is the number of documents in s which contain t, B the number of documents in d which contain t, C the number of documents in s without t, D the number of documents in d without t, and N is the total number of examined documents. The number of documents refers to the document frequency from Section 3.2. The n terms with highest $\chi^2(t,s)$ values that occur more frequently in s than in d are selected because they are least independent from s and are thus best suited to discriminate the most similar tracks from the most dissimilar. The selected terms for each track are then joined into a global list. Throughout the experiments we use $n \in 50, 100, 150$ resulting in reduced features spaces with dimensionality of 4,679, $6,975$, and $8,866$, respectively.

The applicability of other methods for dimensionality reduction, in particular techniques like *probabilistic Latent Semantic Indexing* [13], will be assessed in future work. Considering the size of the evaluation collection and the dimensionality of the unpruned term vector space, for now, we decided to design and implement a method involving only moderate computational expenses, i.e. linear complexity in the number of tracks.

3.5 Vector Adaptation

To further exploit the additional information of the audio similarity measure, we use acoustically similar pieces to adapt the term vector representations of pieces. This is particularly necessary for tracks where no related information could be retrieved from the Web. For all other tracks, the intention is to enforce those characteristics, i.e. those dimensions, that are typical among acoustically similar tracks. To accomplish this, we perform a simple smoothing of the term vector weights depending on the similarity rank. The first vector adaptation approach uses a Gauss weighting. Modified weights of term t for music piece m are defined as

$$gauss_n(t,m) = \sum_{i=0}^{n} \frac{1}{\sqrt{2\pi}} e^{-\frac{(i/2)^2}{2}} \cdot w(t, sim_i(m)), \tag{4}$$

where $sim_i(m)$ is the ith most similar track to m according to audio similarity and $sim_0(m)$ is m itself. The second vector adaptation we examine is a simple linear weighting:

$$linear_n(t,m) = \sum_{i=0}^{n} (1 - \frac{i}{n}) \cdot w(t, sim_i(m)), \tag{5}$$

After adapting the term weights, vectors are again Cosine normalized. In Section 4, we will study the impact of re-weighting vectors based on the 5 and 10 nearest neighbors on the retrieval process.

3.6 Querying the Music Search Engine

After constructing a term vector representation for each piece in the music collection, we need a method to find those tracks that are most similar to a natural language query. The most simple approach would be to map the terms from the query to the respective dimensions and to calculate a score for each track over all query terms. While this could be performed very quickly, the problem is that the possible query terms would be restricted to the vocabulary of the feature space. Hence, to obtain a vector space representation also for queries that consist of terms not in the vocabulary, we extend queries to the music search engine by the word *music* and send them to Google.[9] For reasons of efficiency and performance, only the 10 top-most Web pages are downloaded. Using these pages, a query vector is constructed in the feature space that can now be compared to the music pieces in the collection by calculating *Euclidean distances* on the Cosine normalized vectors. Based on the distances, a *relevance ranking* can be obtained which forms the response to the query.

However, while this approach allows for a virtually unlimited number of queries, it is again dependent on the availability of Google, i.e. to query a local database, the Internet must be accessible. Moreover, the response time of the system increases by the time necessary to perform the on-line retrieval. Furthermore, it is questionable whether only 10 pages are sufficient for proper query expansion. We will address these issues in future work, for which we plan to build an index of all Web documents retrieved during the Web-feature extraction. Query vectors could then be constructed quickly based on this off-line index. Additionally, the comparison of the query vector to all feature vectors in the collection is very inefficient. This could be improved by implementing more sophisticated indexing and search algorithms, e.g. [10].

4. EVALUATION

In this section, the performance of our music search engine approach is evaluated. For testing, we compiled a large music collection of mp3 files from the personal collections of the authors. We also added about 1,000 tracks from the on-line music reseller Magnatune[10] – which are freely available for non-commercial use – to include tracks from not that well-known artists. In total, the complete collection consists of

[9] During evaluation, we also add the constraint *-site:last.fm*.

[10] `http://www.magnatune.com`

14,342 mp3 files from which 12,601 remain after the preprocessing step (Section 3.1). The test set comprises tracks by 1,200 artists and 2,073 albums (with samplers and compilations counted as different albums for each artist). Thus, to obtain Web-based features for this collection, 15,874 different queries to Google are necessary.

4.1 Audioscrobbler Ground Truth

Evaluating the quality of a retrieval system for music is a non-trivial task. The most common approach is to evaluate the results against genre information. Beside the labor intensive task of manually defining genres for a collection of more than 12,000 pieces, this method has several drawbacks, cf. [19]: First, in most cases musical pieces can not be assigned to just one genre. Second, judgments whether a piece belongs to a genre or not are highly subjective. Finally, and most important, it is unclear how to match arbitrary queries to genre information. Since it is our goal to develop a natural language search engine for music, we have to evaluate it on "real-world" queries. Hence, we need a source for phrases which are used by people to describe music and which are likely to be used when searching for music. As mentioned before, we utilize the track specific tag information provided by Last.fm for evaluation. For convenience, Audioscrobbler[11] provides Web Services to access Last.fm data in a standardized manner. From our collection of 12,601 tracks, we find 7,148 to have Audioscrobbler tags assigned (56.7%). For 4,903 of the remaining tracks, tag information about the corresponding artist is available. Thus, in sum we can use 12,051 tracks for performance measurement. The remaining 550 tracks are ignored during the evaluation. Taking a look at the associated tags reveals that they are highly inconsistent and noisy. Many users use personalized identifiers to track their music collection, e.g. by using their nickname as tags for tracks they own. Frequently, tags contain typos, are redundant (e.g. HipHop - Hip Hop, or Hardrock - Hard Rock - Harder Rock), or simply not useful for our purpose ("favorite artist", "mistagged" etc.). However, for lack of a real golden standard, using Last.fm tags is still the best choice. In total, we could observe 43,472 different tags for our collection. Performing evaluation on this complete set is neither reasonable nor feasible. Instead, we make use of the available list of top tags for tracks[12] which contains the 249 most frequently used tags to describe tracks. From this list we remove useless entries like "music", "misc", "albums i own", "test1" etc., entries starting with "my" or "favorite", and one entry that never occurs in our collection. The remaining 227 tags are used as test queries for our system. All music pieces labeled with the query tag are considered relevant wrt. the query. Table 2 shows the most and least frequent tags in our collection.

4.2 Performance Evaluation

Using the Audioscrobbler ground truth, we evaluate the performance of the system with different parameters. Our goal is to study the impacts of the dimensionality of the feature space (cf. Section 3.4) and the vector modifications based on the audio similarity (cf. Section 3.5) on the retrieval quality. In the following, we use the notation χ^2/n to describe the strategy of selecting n most discriminating terms per track in the term selection step. In the experiments we use $n = 50, 100, 150$, resulting in reduced features spaces of dimensionality 4,679, $6,975$, and $8,866$, respectively. Furthermore, the impact of *gauss* and *linear* weighting is examined (for both weightings, we evaluate usage of 5 and 10 neighbors).

label	*tracks labeled*
rock	8,723 (73%)
alternative	8,166 (68%)
seen live	8,067 (67%)
pop	7,953 (66%)
indie	7,517 (63%)
electronic	6,463 (54%)
...	
game music	29 (0.2%)
icelandic	29 (0.2%)
post hardcore	21 (0.2%)
korean	8 (0.1%)
no top tag label	200 (1.7%)

Table 2: Most and least frequent Audioscrobbler top tags in our collection.

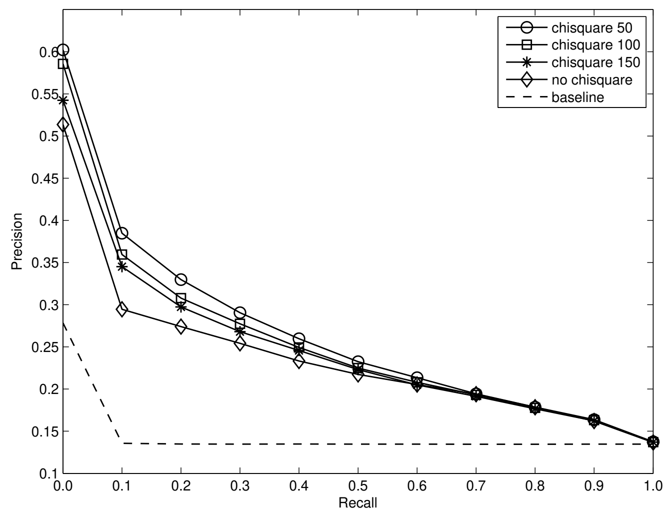

Figure 1: Precision at 11 standard recall levels (average over all 227 test queries) for different term selection settings on Web features without audio-based adaptations.

To measure the quality of the obtained rankings, we calculate standard evaluation measures for retrieval systems, cf. [4]. Figure 1 shows the *Precision at 11 standard recall levels* for the modified *tf* $\times$ *idf* vectors according to Equation 1 (without audio-based re-weighting) for the different χ^2/n settings. This measure is useful to observe precision over the course of a ranking. Since we evaluate the system using a set of 227 queries, we calculate the average of the precision values at each recall level after interpolating to the 11 standard values. It can be seen that audio-based term selection has a very positive impact on the retrieval. The

[11] `http://www.audioscrobbler.net`

[12] `http://ws.audioscrobbler.com/1.0/tag/toptags.xml`

Recall level	0	10	20	30	40	50	60	70	80	90	100
$tf \times idf$	60.21	38.47	32.97	29.05	**25.97**	**23.24**	**21.34**	**19.43**	**17.86**	**16.38**	13.75
$gauss, n = 5$	61.96	**38.49**	33.29	29.01	25.13	22.51	20.60	18.99	17.38	15.89	13.83
$gauss, n = 10$	**62.15**	38.47	**33.31**	**29.08**	25.15	22.52	20.60	18.98	17.39	15.89	13.83
$linear, n = 5$	60.49	37.58	32.40	28.33	24.79	22.16	20.24	18.74	17.21	15.81	13.81
$linear, n = 10$	58.09	37.38	32.41	28.34	24.74	22.25	20.22	18.67	17.21	15.82	**13.86**

Table 3: *Precision at 11 standard recall levels* for $\chi^2/50$ in percent (average over all 227 test queries).

setting $\chi^2/50$ yields best results, while term vectors without dimensionality reduction perform worst. In Figure 1, one can also see the baseline of all experiments, which has been empirically determined by repeated evaluation of random permutations of the music collection. All results obtained with our approach are far above the baseline.

To give stronger evidence for the effects of the vector space pruning, we also perform statistical significance tests. Since the precision values obtained for the different queries are not normally distributed, we have to apply a non-parametric test. To compare all different settings "simultaneously", we use Friedman's test (pairwise significance tests would drastically increase the probability of an alpha-error). For lack of a test that takes the development of precision values over the different recall levels into account, we perform a Friedman test at each separate recall level (with attached post-hoc tests; significance level = 0.05). At recall levels 0.0 to 0.4, the $\chi^2/50$ setting is significantly better than both $\chi^2/150$ and *no* χ^2, at recall levels 0.1 to 0.3, $\chi^2/50$ is even significantly better than $\chi^2/100$. For the remaining recall levels (0.5 to 1.0), no reasonable significant difference between the settings can be found.

While audio similarity-based term selection has an evident positive impact on the retrieval quality, the impact of audio-based vector re-weighting is only marginal. For reasons of lucidity, the precision values at 11 standard recall levels for $tf \times idf$, $gauss$, and $linear$ vectors (all $\chi^2/50$) are shown in Table 3 instead of a graph (the curves would be too similar to gain insights). At the 0.0 recall level we can (theoretically) expect precision values around 0.6, at the 0.1 level, precision drops to about 0.4. A deeper look into the results of individual queries reveals that especially sparsely distributed tags perform bad (cf. Table 2). The fact that queries with terms like *"korean"* result in very low precision is no surprise, since in reality there are no korean music pieces in the collection (although the Audioscrobbler tags suggest this).

Again, we perform Friedman tests to check for significant differences between the various re-weighting strategies. The mean values in Table 3 conceal the relations of the underlying distributions to some extent. Using the Friedman test, it can be seen that $tf \times idf$ performs significantly worse than $gauss_{n=5}$, $gauss_{n=10}$, and $linear_{n=5}$ at the 0.0 recall level. At the 0.1 and 0.3 level, the $tf \times idf$ values are significantly below $gauss_{n=5}$, $gauss_{n=10}$, and $linear_{n=10}$, at 0.2, $tf \times idf$ is below all others. Between recall levels 0.4 and 0.8 there is no significant difference between $tf \times idf$ and the gauss weighted approaches. In general, we can observe a positive impact of vector re-weighting on the precision at lower recall levels (which are more relevant in practice). However, the achieved improvements are rather small.

Furthermore, we average single value summaries over all queries. Although single value summaries are more meaningful for the individual queries, as pointed out in [4], we still include the results to document the general tendencies. The *average precision at seen relevant documents* indicates the ability of the different settings to retrieve relevant documents quickly (Table 4). The respective values are calculated by averaging the current precision values at each observed relevant document. A similar measure is *R-precision* (Table 5). It corresponds to the precision at the Rth position in the ranking, where R is the number of relevant documents for the query. Finally, we calculate the *precision after 10 documents*. Since returning 10 results is the default for nearly all search engines, we think it is valuable to examine how many relevant music pieces can be expected "at first sight" (Table 6). As can be seen for the best settings, in average, every second piece among the first ten is relevant.

feature space	$\chi^2/50$	$\chi^2/100$	$\chi^2/150$	no χ^2
$tf \times idf$	**25.63**	**24.52**	**23.93**	**22.66**
$gauss, n = 5$	25.28	24.24	23.67	22.33
$gauss, n = 10$	25.29	24.25	23.68	22.33
$linear, n = 5$	24.78	23.77	23.24	21.91
$linear, n = 10$	24.74	23.72	23.22	21.79

Table 4: *Average precision at seen relevant documents* for different settings in percent (average over all 227 test queries).

feature space	$\chi^2/50$	$\chi^2/100$	$\chi^2/150$	no χ^2
$tf \times idf$	26.07	25.16	24.66	**23.93**
$gauss, n = 5$	26.40	25.50	24.94	23.78
$gauss, n = 10$	**26.41**	**25.51**	**24.95**	23.77
$linear, n = 5$	25.99	25.19	24.67	23.37
$linear, n = 10$	26.07	25.20	24.72	23.16

Table 5: *R-precision* for different settings in percent (average over all 227 test queries).

feature space	$\chi^2/50$	$\chi^2/100$	$\chi^2/150$	no χ^2
$tf \times idf$	**49.69**	45.46	44.19	**40.09**
$gauss, n = 5$	49.60	**46.12**	**45.37**	38.55
$gauss, n = 10$	49.56	46.08	44.98	38.77
$linear, n = 5$	47.67	44.71	43.66	36.70
$linear, n = 10$	47.00	44.23	42.33	36.04

Table 6: *Precision after 10 documents* for different settings in percent (average over all 227 test queries).

4.3 Examples

Finally, we want to present some search results from our test collection. Table 7 shows the top 10 music pieces for the queries *"rock with great riffs"*, *"punk"*, and *"relaxing music"*. For *"rock with great riffs"*, all relevant music pieces are from the same artist (other artists with great riffs follow in the ranking). "Easy" queries with relatively clearly defined musical styles like *"punk"* work very well. A query like *"relaxing music"* is difficult to judge because it is in fact very subjective. However, the majority of the returned pieces is the "opposite of exciting" and would probably be considered to be "relaxing" by many people. Interestingly, 9 out of 10 results are published by the label Magnatune.

Beside finding adequate music for descriptive queries, the system offers some simple but useful features for free. Based on the related Web pages, "semantic relations" are often implicitly available for queries. For example, the query *"Damon Albarn"* returns tracks from the band "blur", as well as from the band "Gorillaz" (both bands of Damon Albarn). Similarly, searching for *"George Harrison"* returns also tracks from "The Beatles".

	Search results for *'rock with great riffs'*
1.	**Wolfmother – Vagabond**
2.	**Wolfmother – Colossal**
3.	**Wolfmother – Tales**
4.	**Wolfmother – Love train**
5.	**Wolfmother – Woman**
6.	**Wolfmother – White unicorn**
7.	Clap your hands say yeah – Heavy metal
8.	**Wolfmother – Mind's eye**
9.	The Chemical Brothers – Let forever be
10.	**Wolfmother – Witchcraft**

	Search results for *'punk'*
1.	**Buzzcocks – Breakdown**
2.	**New York Dolls – Seven day weekend**
3.	**Sham 69 – Angels with dirty faces**
4.	**New York Dolls – Subway train**
5.	**Sham 69 – Hersham boys**
6.	**Buzzcocks – Orgasm addict**
7.	**New York Dolls – Looking for a kiss**
8.	**Screeching Weasel – Someday**
9.	**Buzzcocks – Love battery**
10.	**Electric Frankenstein – Home of the brave**

	Search results for *'relaxing music'*
1.	**Jamie Janover – Innerlude**
2.	Jamie Janover – Two plus blue
3.	**Michael Masley – Six white horses**
4.	**Jamie Janover – Twice versa**
5.	**Jamie Janover – Ragasutra**
6.	**Psychetropic – Frozen garden**
7.	Aerobic Jonquil – Click full pussy
8.	**Jean-Michel Jarre – Magnetic fields 1**
9.	Cargo Cult – Ambriel
10.	Solace – Circle

Table 7: Retrieval results for three queries. Bold entries indicate relevant music pieces.

5. CONCLUSIONS AND FUTURE WORK

We presented a first attempt to create a search engine for large music collections that can be queried via natural language text input. One of the central challenges of our method is to assign semantically related information to individual music pieces. We opt to accomplish this by finding relevant information on the Web. The extracted text based information is complemented by audio-based similarity, which allows us to improve the results of the retrieval by reducing the dimensionality of the feature space. Information about the acoustic similarity is also mandatory to describe music pieces for which no related pages can be found on the Web.

Estimating the quality of the presented approach is difficult since there exists no related approach that could serve as reference. Nevertheless, compared to the baseline, the results we obtained during evaluation were very promising. Considering the fact that the used ground truth has severe drawbacks, results can be interpreted as a "conservative" estimation of the real performance. However, there is ample space for improvements. In future work, we will elaborate methods to construct Web-based feature vectors more efficiently. Furthermore, we believe that there is much more potential in integrating audio-based similarity, especially if improved audio similarity measures become available. Future enhancements will also comprise special treatment of terms appearing in the meta-tags of the mp3 files and the search for phrases in lyrics. Finally, we can state that we are confident that these first steps point into the right direction and that a natural language search engine for music is feasible.

6. ACKNOWLEDGMENTS

This research is supported by the Austrian Fonds zur Förderung der Wissenschaftlichen Forschung (FWF) under project number L112-N04 and the Vienna Science and Technology Fund (WWTF), project CIO10 "Interfaces to Music". The Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Ministry for Education, Science, and Culture and by the Austrian Federal Ministry for Transport, Innovation, and Technology.

7. REFERENCES

[1] J.-J. Aucouturier. *Ten Experiments on the Modelling of Polyphonic Timbre*. PhD thesis, University of Paris 6, 2006.

[2] J.-J. Aucouturier and F. Pachet. Music Similarity Measures: What's the Use? In *Proceedings of the 3rd International Conference on Music Information Retrieval*, Paris, France, 2002.

[3] J.-J. Aucouturier, F. Pachet, and M. Sandler. "The Way It Sounds": Timbre Models for Analysis and Retrieval of Music Signals . *IEEE Transactions on Multimedia*, 7(6):1028–1035, December 2005.

[4] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, Reading, Massachusetts, 1999.

[5] S. Baumann, A. Klüter, and M. Norlien. Using natural language input and audio analysis for a human-oriented mir system. In *Proceedings of the 2nd International Conference on Web Delivering of Music*, Darmstadt, Germany, 2002.

[6] P. Cano, M. Koppenberger, N. Wack, et al. An industrial-strength content-based music recommendation system. In *Proceedings of 28th Annual International ACM SIGIR Conference*, Salvador, Brazil, 2005.

[7] I. Celino, E. Della Valle, D. Cerizza, and A. Turati. Squiggle: a semantic search engine for indexing and retrieval of multimedia content. In *Proceedings of the 1st International Workshop on Semantic-enhanced Multimedia Presentation Systems*, Athens, Greece, 2006.

[8] O. Celma, P. Cano, and P. Herrera. Search Sounds: An audio crawler focused on weblogs. In *Proceedings of the 7th International Conference on Music Information Retrieval*, Victoria, B.C., Canada, 2006.

[9] O. Celma, M. Ramirez, and P. Herrera. Foafing the music: A music recommendation system based on RSS feeds and user preferences. In *Proceedings of the 6th International Conference on Music Information Retrieval*, London, UK, 2005.

[10] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, San Francisco, CA, USA, 1997.

[11] A. Ghias, J. Logan, D. Chamberlin, and B. C. Smith. Query by humming: musical information retrieval in an audio database. In *Proceedings of the 3rd ACM International Conference on Multimedia*, San Francisco, CA, USA, 1995.

[12] M. Goto and T. Goto. Musicream: New Music Playback Interface for Streaming, Sticking, and Recalling Musical Pieces. In *Proceedings of the 6th International Conference on Music Information Retrieval*, London, UK, 2005.

[13] T. Hofmann. Probabilistic Latent Semantic Indexing. In *Proceedings of 22nd Annual International ACM SIGIR Conference*, Berkeley, CA, USA, 1999.

[14] P. Knees, E. Pampalk, and G. Widmer. Artist Classification with Web-based Data. In *Proceedings of 5th International Conference on Music Information Retrieval*, Barcelona, Spain, 2004.

[15] P. Knees, M. Schedl, T. Pohle, and G. Widmer. An Innovative Three-Dimensional User Interface for Exploring Music Collections Enriched with Meta-Information from the Web. In *Proceedings of the 14th ACM International Conference on Multimedia*, Santa Barbara, CA, USA, 2006.

[16] I. Knopke. Aroooga: An audio search engine for the world wide web. In *Proceedings of the International Computer Music Conference*, Miami, USA, 2004.

[17] N. C. Maddage, H. Li, and M. S. Kankanhalli. Music structure based vector space retrieval. In *Proceedings of 29th Annual International ACM SIGIR Conference*, Seattle, USA, 2006.

[18] M. Mandel and D. Ellis. Song-Level Features and Support Vector Machines for Music Classification. In *Proceedings of the 6th International Conference on Music Information Retrieval*, London, UK, 2005.

[19] F. Pachet and D. Cazaly. A Taxonomy of Musical Genres. In *Proceedings of RIAO 2000 Content-Based Multimedia Information Access*, Paris, France, 2000.

[20] E. Pampalk. *Computational Models of Music Similarity and their Application to Music Information Retrieval*. PhD thesis, Vienna University of Technology, March 2006.

[21] E. Pampalk and M. Goto. Musicrainbow: A new user interface to discover artists using audio-based similarity and web-based labeling. In *Proceedings of the 7th International Conference on Music Information Retrieval*, Victoria, B.C., Canada, 2006.

[22] T. Pohle, P. Knees, M. Schedl, and G. Widmer. Automatically Adapting the Structure of Audio Similarity Spaces. In *Proceedings of the 1st Workshop on Learning the Semantics of Audio Signals*, Athens, Greece, 2006.

[23] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[24] B. Whitman. *Learning the meaning of music*. PhD thesis, Massachusetts Institute of Technology, 2005.

[25] B. Whitman and S. Lawrence. Inferring Descriptions and Similarity for Music from Community Metadata. In *Proceedings of the 2002 International Computer Music Conference*, Gothenburg, Sweden, 2002.

[26] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the 14th International Conference on Machine Learning*, Nashville, USA, 1997.