Searching for Music Using Natural Language Queries and Relevance Feedback

Peter Knees[1] and Gerhard Widmer[1,2]

[1] Dept. of Computational Perception, Johannes Kepler University Linz, Austria
[2] Austrian Research Institute for Artificial Intelligence (OFAI)
peter.knees@jku.at

Abstract. We extend an approach to search inside large-scale music collections by enabling the user to give feedback on the retrieved music pieces. In the original approach, a search engine that can be queried through free-form natural language text is automatically built upon audio-based and Web-based similarity measures. Features for music pieces in the collection are derived automatically by retrieving relevant Web pages via Google queries and using the contents of these pages to construct *term vectors*. The additional use of information about acoustic similarity allows for reduction of the dimensionality of the vector space and characterization of audio pieces with no associated Web information. With the incorporation of *relevance feedback*, the retrieval of pieces can be adapted according to the preferences of the user and thus compensate for inadequately represented initial queries. The approach is evaluated on a collection comprising about 12,000 pieces by using semantic tags provided by *Audioscrobbler* and a user study which also gives further insights into users search behaviors.

1 Introduction

When searching for (popular) music, users' options are currently very limited. Existing music search systems, i.e. the search systems offered by commercial music portals, make use of meta-data like artist, album name, track name, or year as well as arbitrarily determined, subjective meta-information like genre or style to index their music repository. As a consequence, when searching for music, the customer must already have a very precise conception of the expected result set. In fact, none of these systems allows its users to formulate natural language queries that *describe* the music they intend to find. For example, instead of just finding tracks that are assigned to the genre Rock, a user could want to formulate a query like *"rock with great riffs"* to emphazise the importance of energetic guitar phrases in the desired music pieces. Another example would be the query *chicago 1920*, which could express the intention to find Jazz pieces originating from this particular area and time.

To address the shortcomings of current search systems, in [11] we proposed a method to build a music search engine that is capable of processing arbitrary queries. For each piece in a music collection, features are derived automatically from relevant Web pages by constructing *term vector* representations using standard Information Retrieval methods. Furthermore, a state-of-the-art audio similarity measure is incorporated to characterize audio pieces with no (or little) Web information associated. Thus, we combine information about the *context* of music with information about the *content*.

However, although the musical, or more general, the cultural context of music pieces can be captured to a certain extent with this method, there are still limitations. One obvious problem is the appropriate translation of queries into the term vector space of music pieces to calculate similarities to all retrievable pieces. Furthermore, we have to deal with the fact that users are actually not accustomed to use free-form text input to search for music. Even if these issues can be sorted out in the near future, the problem of individual concepts and intentions behind the issued queries remains. For example, different users will have different expectations of the resulting pieces for the query *folk*. Some users may aim to retrieve music pieces from american singers and songwriters, while others may intend to find all sorts of folkloristic music. While these user specific interests may not be adequately expressible via a query, getting explicit feedback on the relevance of the retrieved pieces from the users can give extremely valuable information to disambiguate query meaning and clarify the original intention.

In this paper, we incorporate Rocchio's relevance feedback method to adapt the retrieval of music pieces to the user's preferences. Not only that the retrieval process can increasingly accommodate to users expectations, the approach can also help to compensate for inadequately translated initial queries that would otherwise result in low performance.

2 Related Work

In the following, we review music information retrieval systems that enable cross-media retrieval, i.e. 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. Compared to the number of presented *query-by-example* systems (e.g. [13, 8]), the number of systems allowing for this form of query is very little. Beside our own approach [11], the most elaborate work has been presented in [5]. The system is supported by a semantic ontology which integrates meta-data as well as automatically extracted acoustic properties of the music pieces and defines relations between these informations. In the end, the system allows for semantic queries like *"something fast from..."* or *"something new from..."*. In [7], the music search engine *Search Sounds*[3] is presented. A special crawler that focuses on a set of "audio blogs" is used to find blog entries consisting of music files with associated explanations. The related textual information can then be used to match text queries to actual music pieces. Furthermore, acoustically similar pieces can be discovered by means of content-based audio analysis. Another system that opts

[3] `http://www.searchsounds.net`

to enhance music search with additional semantic information is Squiggle [6]. Queries are matched against meta-data provided from the freely available community databases MusicMoz[4] and MusicBrainz[5]. Based on this data, related queries are proposed, for example, searching for *rhcp* results in zero hits, but suggests to search for the band "Red Hot Chili Peppers".

A system that is not limited to a fixed set of predefined meta-data is the recommendation service Last.fm[6]. Last.fm monitors each user's listening preferences by integrating into music player applications. Based on the collected data, similar artists or tracks are identified and can be recommended to other users. 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 the assigned tags are highly inconsistent and noisy, cf. [11].

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 [12] and FindSounds[7].

3 Technical Realization

In Sections 3.1 to 3.5, we review our technique to build a natural language search engine for music as described in [11]. Instead of describing every step in detail, we will solely present the settings that yielded best results during evaluation. In Section 3.6, we describe the straight-forward incorporation of Rocchio's relevance feedback technique into our system.

3.1 Web-based Features

We rely on the Web as our primary source of information. While previous work exploiting Web data for Music Information Retrieval operates solely on the artist level (e.g. [19, 10]), we try to derive descriptors for individual tracks. To this end, we opt to gather as much track specific information as possible while preserving a high number of available web pages (via artist related pages) by joining 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

[4] `http://www.musicmoz.org`
[5] `http://www.musicbrainz.org`
[6] `http://www.last.fm`
[7] `http://www.findsounds.com`

While the first query is intended to provide a stable basis of artist related documents, the second and third query target more specific pages (reviews of the album and the track, respectively). For each query, at most 100 of the top-ranked Web pages are retrieved and added to the set of pages relevant to the track. 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 tf_{tm}, the number of occurrences (term frequency) of term t in documents related to m, as well as df_{tm}, the number of pages related to m in which the term t occurred (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. For our evaluation collection, this results in a vector space with 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 [18]:

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

where N is the overall number of music pieces in the collection. From the given definition, it can be seen that all Web pages related to a music piece are treated as one large document. Furthermore, the resulting term weight vectors are Cosine normalized to remove the influence of the length of the retrieved Web pages as well as the different numbers of retrieved pages per track.

3.2 Audio-based Similarity

In addition to the context-based Web features, information on the content of the music is derived by following a well-established procedure, e.g. [3, 15]: For each audio track, 19 Mel Frequency Cepstral Coefficients (*MFCCs*) are computed on short-time audio segments (called *frames*) to describe the spectral envelope of each frame. Thus, perceived acoustical similarity is assessed by modeling *timbral* properties. For MFCC calculation, we use the definition given in [2]:

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

According to [14], a Single Gaussian Model with full covariance matrix is sufficient to model the distribution of MFCCs. This facilitates computation and comparison of the distribution models, since a symmetrized *Kullback-Leibler divergence* can be calculated on the means and covariance matrices in order to derive a similarity measure.

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

However, applying the Kullback-Leibler divergence entails some undesirable consequences [1, 16], e.g., it can be observed that some pieces ("hubs") are frequently "similar" (i.e. have a small distance) to many other pieces in the collection without actually sounding similar, while on the other side, some pieces are never similar to others. Furthermore, the Kullback-Leibler divergence is no metric since it does not fulfill the triangle inequality. To deal with these issues, we apply a simple rank-based correction called *Proximity Verification* [16]. As a consequence, all further steps presented here will be based on the ranking information of the audio similarity measure only.

3.3 Dimensionality Reduction

For dimensionality reduction of the feature space, we use the χ^2 test, e.g. [20]. Since we have no class information (e.g. genre) available, 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 100 most similar tracks, from d, the group of the 100 most dissimilar tracks. 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 term 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 the total number of examined documents. The number of documents refers to the document frequency from Section 3.1. We found to yield best results when joining into a global list the 50 terms of each track's calculation that have highest $\chi^2(t, s)$ values and occur more frequently in s than in d. After feature selection, for our collection, 4,679 dimensions remain.

3.4 Vector Adaptation

Another application of the information provided by the audio similarity measure is the modification of the term vector representations toward acoustically similar pieces. This step is mandatory for tracks for which no related information could be retrieved from the Web. For all other tracks, the intention is to enforce those dimensions that are typical among acoustically similar tracks. To this end, a simple Gauss weighting over the $n = 10$ most similar tracks is performed for each piece. 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 $sim_i(m)$ denotes the i^{th} most similar track to m according to audio similarity and $sim_0(m)$ is m itself. Vectors are again Cosine normalized after term weight adaptation.

3.5 Querying the Music Search Engine

Finding those tracks that are most similar to a natural language query is a non trivial task. In [11], queries are translated to vector space representations by adding the extra constraint *music*, sending them to Google and constructing a term vector from the 10 top-most Web pages returned. The resulting query vector can then 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.

This method has two major drawbacks. First, it depends on the availability of Google, i.e. to query the local database, the Internet must be accessible, and second, the response time of the system increases by the time necessary to perform the on-line retrieval. To by-pass these shortcomings, we utilize the Web pages retrieved for term vector creation to create an off-line index that can be used instead of Google. For our purpose, we configured the Java-based open source search engine *Nutch*[9] to index the off-line collection of documents. Since information on in- and out-links is not available for the stored documents, *Nutch* calculates the document relevances for a query based on $tf \times idf$ values.[10]

3.6 Relevance Feedback

Relevance feedback is an iterative process, in which the user is presented with a ranked list of the music pieces that are most similar to the query. After examination of the list, the user marks those pieces which are relevant in his/her opinion (*explicit relevance feedback*).[11] The intention is to modify the query vector such that it moves toward the relevant and away from the non-relevant pieces. Since both music pieces and queries are representable as weighted term vectors, we can easily incorporate Rocchio's relevance feedback method to adapt search results according to users' preferences [17]. Thus, based on the relevance judgments, we calculate the modified query vector $\vec{q}_m$ by (cf. [4])

$$\vec{q}_m = \alpha\,\vec{q} + \frac{\beta}{|D_r|}\sum_{\forall \vec{d}_j \in D_r} \vec{d}_j - \frac{\gamma}{|D_n|}\sum_{\forall \vec{d}_j \in D_n} \vec{d}_j, \tag{5}$$

where $\vec{q}$ is the original query vector constructed from the stored Web pages, D_r the set of relevant music pieces (according to the user) among the retrieved pieces, and D_n the set of non-relevant pieces among the retrieved pieces. The parameters α, β, and γ can be used to tune the impacts of original vector, relevant pieces, and non-relevant pieces, respectively. For our experiments, we decided to assign equal values to all parameters, i.e. $\alpha = \beta = \gamma = 1$. The modified vector

[9] `http://lucene.apache.org/nutch/`

[10] Note that due to time constraints, only a subset (approx. 100,000 documents) of the stored files (consisting solely of the pages returned for the *"artist" music* queries) has been indexed.

[11] For future work also implicit relevance feedback could be deployed by measuring e.g. the time a user is listening to the returned tracks.

is again Cosine normalized. Based on the new query vector, new results are presented to the user in the next step. The effect of relevance feedback (modification after 20 pieces) can be seen in Table 4 for the example query *speed metal*.

Note that, in contrast to information retrieval systems for text documents that often benefit from additional techniques like *query expansion* [9], our system is currently restricted to the query vector modification step due to the two-layered query processing.

4 Evaluation

In this section, the performance of our extended music search engine is evaluated. For reasons of comparability, we evaluate the impact of relevance feedback on the evaluation collection from [11]. The collection comprises 12,601 different tracks by 1,200 artists. Evaluation is carried out on the same set of related Web pages using the same semantic tags. Additionally, we report on a user study that has been conducted to uncover the impact of relevance feedback.

4.1 Evaluation against Audioscrobbler Ground Truth

As in [11], we utilize the track specific tag information provided by Last.fm/Audioscrobbler for evaluation. Although this method has severe shortcomings (for a discussion see [11]), for lack of a real golden standard, using Last.fm/Audioscrobbler tags is still a viable option. The same set of 227 test queries is used to evaluate the performance of our system. As reference values, we include the best performing method from [11] which was obtained by pruning the vector space to 4,679 dimensions, Gaussian smoothing over ten nearest neighbors and query construction by invoking Google.

To measure the impact of relevance feedback, for each test query we construct two rankings. The first is the standard ranking for the whole collection based on a query vector constructed from the offline-index of Web pages. For the second ranking, we simulate relevance feedback by starting with the first 20 results obtained through an off-line based query vector. The next 20 results are then calculated from the query vector modified according to the relevances of the already seen music pieces, and so on.

We measure the quality of the rankings obtained with the different retrieval approaches by calculating the *precision at 11 standard recall levels* for the three compared retrieval methods, cf. [4]. 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. The resulting plots are depicted in Figure 1. Not surprisingly, the usage of relevance feedback has a very positive effect on the precision of the returned music pieces. Starting from the same level (about 0.49 precision at recall level 0.0) the traditional approach without relevance feedback drops to 0.34 precision at recall level 0.1, while relevance feedback boosts precision to 0.52. Also for all other recall levels this trend is clearly visible. Beside

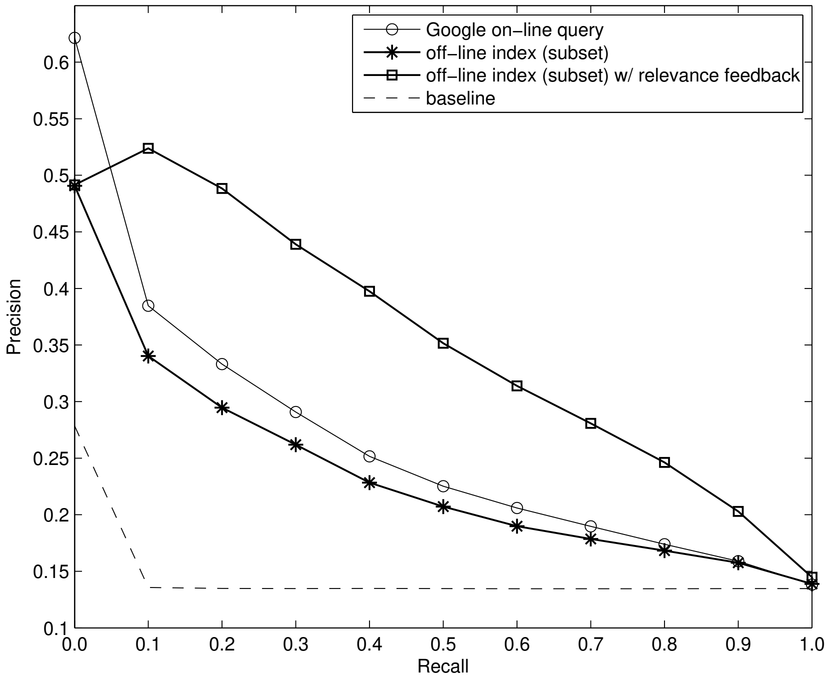

Fig. 1. Precision at 11 standard recall levels (average over all 227 test queries) for the different retrieval approaches.

this, it can also be seen that the values of the off-line index approach without relevance feedback are thoroughly below the values of the on-line approach that uses Google for query vector construction.

For further comparisons, we average single value summaries over all queries. The *average precision at seen relevant documents* indicates the ability of the different settings to retrieve relevant documents quickly. A similar measure is *R-Precision*. It corresponds to the precision at the Rth position in the ranking, where R is the number of relevant documents for the query. For both measures, the approach utilizing relevance feedback yields the highest values. 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". This setting is only meaningful to compare the on-line Google query approach and the off-line index approaches. As expected, Google performs better here (about every second piece among the first ten is relevant in average). Using the off-line index, in average 4 returned music pieces among the first ten are relevant. This complies with the results obtained by means of a user study presented in the next section.

	Google query	off-line query	off-line/RF
Avg. prec. at seen relevant docs	25.29	22.99	**35.80**
R-Precision	26.41	23.48	**37.66**
Precision after 10 documents	**49.56**	39.74	39.74

Table 1. IR measures for different retrieval approaches (average over all 227 test queries). The first column shows values obtained by constructing query vectors via Google. The second column displays the values obtained by using the constructed off-line index instead. The third column shows the obtained values for relevance feedback enabled (off-line query).

4.2 Evaluation via User Experiments

We conducted a small user study with 11 participants to assess the impact of the relevance feedback under less artificial conditions. To this end, each participant was asked to submit 5 queries of choice to the system. For each query, in total 100 results, whose relevance to the query had to be judged, were presented in groups of 20 (thus, a run consisted of 5 feedback iterations). Additionally, each query had to be evaluated twice. In one run, the ranking was not influenced by the ratings at all, i.e. the first 100 retrieval results without relevance feedback have been presented in groups of 20. In the other run, relevance feedback was enabled. Thus, the ratings of the documents had a direct influence on the following 20 results. Whether the first or the second run was presented first was chosen randomly for each query to avoid learning effects. Furthermore, the users were told to evaluate two different feedback strategies. The fact that one run included no feedback strategy at all was concealed. The 55 different queries issued by the participants can be found in Table 3.

Since obtaining users' relevance judgments for all pieces in the collection for all queries is infeasible, other measures than those used in Section 4.1 have to be applied to illustrate the impact of relevance feedback, e.g. those proposed in [9]. Table 2 displays the results of the user study. Interestingly, for the first iteration, results are not consistent. Obviously, users have considered different tracks to be relevant in the first and in the second run (even if only very sporadically). Nevertheless, the general trend of better results when using relevance feedback can be observed in the user study.

5 Conclusions and Future Work

We successfully incorporated relevance feedback into 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 leads to improved results of the retrieval due to the reduced dimensionality of the feature space. Information about the acoustic

	iter.1	iter.2	iter.3	iter.4	iter.5	total
No relevance feedback						
relevant retrieved/iter. (mean)	7.13	**5.02**	4.05	3.76	3.71	4.73
relevant retrieved/iter. (sum)	392	**276**	223	207	204	1,302
cumulative relevant retr. (sum)	392	**668**	891	1,098	1,302	1,302
queries with all relevant	9	**3**	3	2	1	0
queries with no relevant	22	**26**	25	28	30	17
With relevance feedback						
relevant retrieved/iter. (mean)	7.22	4.15	**6.47**	**5.73**	**6.18**	**5.95**
relevant retrieved/iter. (sum)	397	228	**356**	**315**	**340**	**1,636**
cumulative relevant retr. (sum)	397	625	**981**	**1,296**	**1,636**	**1,636**
queries with all relevant	8	1	**6**	**4**	**5**	0
queries with no relevant	23	27	**20**	**23**	**22**	**11**

Table 2. Results of the user study over 5 iterations. In each iteration, 55 queries have been evaluated (the maximum achievable number of relevant retrieved pieces for each query is 20; the maximum achievable number per iteration is thus 1,100). For comparison between results obtained with relevance feedback and results obtained without, the more advantageous values are set in bold typeface.

similarity is also mandatory to describe music pieces for which no related pages can be found on the Web. Due to the chosen vector space model, Rocchio's relevance feedback could be integrated smoothly. The conducted evaluations showed that relevance feedback provides a valuable extension to the system in that it adapts to users' preferences.

Since relevance feedback has a positive impact on the system's performance, we can conclude that the vector space representations of the music pieces are well suited to model the similarity between pieces. To further advance the system, the translation of queries into the term vector space has to be improved. Starting with better initial results is also mandatory for the acceptance of the system since people usually judge the quality based on the first results.

Finally, for future work, we plan to create a music-related Web page index on our own with the intention to further improve the applicability of the system by abolishing the dependency on external search engines.

6 Acknowledgments

Special thanks are due to all volunteers that helped evaluating the search engine. This research is supported by the Austrian Fonds zur Förderung der Wissenschaftlichen Forschung (FWF 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.

"plastic band"	jazz
80ies synth pop	latin pop
ac/dc	mass in b minor
acdc	melodic metal with opera singer as front woman
american folk	metal
angry samoans	metallica
barbie girl	ndw
cello	neomedieval music
comedy	new orleans
dancehall	new zork scene
don't dream it's over	no new york
drude	nur die besten sterben jung
eurodance	oldies slow jazz
female electro	postmodern
filmmusik	punk
gangsta	punk rock
german hip hop	rammstein music with strong keyboard
ghost dog	rem
green day	schoenheitsfehler
groove	sicherheitsmann
guitar rock brit pop	soundtrack
happy sound	vienna electro dj
hard rock fast guns'n roses	violin
heavy metal with orchestra	weilheim
herr lehmann	wie lieblich sind deine wohnungen
in extremo live	world
indie rock	zztop
industrial rock trent reznor	

Table 3. 55 queries issued by users during the user study.

References

1. Jean-Julien Aucouturier. *Ten Experiments on the Modelling of Polyphonic Timbre.* PhD thesis, University of Paris 6, 2006.
2. Jean-Julien Aucouturier and François Pachet. Music Similarity Measures: What's the Use? In *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR'02)*, pages 157–163, Paris, France, October 2002. IRCAM.
3. Jean-Julien Aucouturier, François Pachet, and Mark 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. Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval.* Addison-Wesley, Reading, Massachusetts, 1999.
5. Stephan Baumann, Andreas Klüter, and Marie 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 (WEDELMUSIC 2002)*, Darmstadt, Germany, 2002.
6. Irene Celino, Emanuele Della Valle, Dario Cerizza, and Andrea 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 (SEMPS 2006), Athens, Greece, 2006.
7. Oscar Celma, Pedro Cano, and Perfecto Herrera. Search Sounds: An audio crawler focused on weblogs. In *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR'06)*, Victoria, B.C., Canada, 2006.
8. Asif Ghias, Jonathan Logan, David Chamberlin, and Brian C. Smith. Query by humming: musical information retrieval in an audio database. In *Proceedings of the 3rd ACM International Conference on Multimedia (MULTIMEDIA'95)*, San Francisco, California, United States, 1995.
9. Donna Harman. Relevance Feedback Revisited. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR'92)*, Copenhagen, Denmark, 1992.
10. Peter Knees, Elias Pampalk, and Gerhard Widmer. Artist Classification with Web-based Data. In *Proceedings of 5th International Conference on Music Information Retrieval (ISMIR'04)*, pages 517–524, Barcelona, Spain, October 2004.
11. Peter Knees, Tim Pohle, Markus Schedl, and Gerhard Widmer. A Music Search Engine Built upon Audio-based and Web-based Similarity Measures. In *Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval (SIGIR'07)*, Amsterdam, the Netherlands, July 23-27 2007.
12. Ian Knopke. AROOOGA: An audio search engine for the World Wide Web. In *Proceedings of the 2004 International Computer Music Conference (ICMC'04)*, Miami, USA, 2004.
13. Namunu C. Maddage, Haizhou Li, and Mohan S. Kankanhalli. Music structure based vector space retrieval. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, 2006.
14. Michael Mandel and Dan Ellis. Song-Level Features and Support Vector Machines for Music Classification. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR'05)*, London, UK, 2005.
15. Elias Pampalk. *Computational Models of Music Similarity and their Application to Music Information Retrieval*. PhD thesis, Vienna University of Technology, March 2006.
16. Tim Pohle, Peter Knees, Markus Schedl, and Gerhard Widmer. Automatically Adapting the Structure of Audio Similarity Spaces. In *Proceedings of 1st Workshop on Learning the Semantics of Audio Signals (LSAS 2006)*, Athens, Greece, December 2006.
17. Joseph J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System – Experiments in Automatic Document Processing*. Prentice Hall Inc., Englewood Cliffs, NJ, USA, 1971.
18. Gerard Salton and Chris Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.
19. Brian Whitman and Steve Lawrence. Inferring Descriptions and Similarity for Music from Community Metadata. In *Proceedings of the 2002 International Computer Music Conference (ICMC'02)*, pages 591–598, Gotheborg, Sweden, September 2002.
20. Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In D. H. Fisher, editor, *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 412–420, Nashville, US, 1997. Morgan Kaufman Publishers, San Francisco, US.

	no relevance feedback	with relevance feedback
1.	Deicide - Dead But Dreaming	Deicide - Dead But Dreaming
2.	Deicide - Trifixion	Deicide - Trifixion
3.	Deicide - Repent To Die	Deicide - Repent To Die
4.	Skitzo - Kill With a Vengeance (live)	Skitzo - Kill With a Vengeance (live)
5.	Deicide - In Hell I Burn	Deicide - In Hell I Burn
6.	**Iron Savior - Protector**	**Iron Savior - Protector**
7.	Entombed - Chief Rebel Angel	Entombed - Chief Rebel Angel
8.	Deicide - Satan Spawn, The Caco-Daemon	Deicide - Satan Spawn, The Caco-Daemon
9.	**Iron Savior - Warrior**	**Iron Savior - Warrior**
10.	**Nightwish - Nightshade Forests**	**Nightwish - Nightshade Forests**
11.	**Powergod - Back To Attack**	**Powergod - Back To Attack**
12.	Deicide - Oblivious To Evil	Deicide - Oblivious To Evil
13.	**Steel Prophet - Unseen**	**Steel Prophet - Unseen**
14.	**Steel Prophet - The Ides Of March**	**Steel Prophet - The Ides Of March**
15.	**Steel Prophet - Messiah**	**Steel Prophet - Messiah**
16.	**Steel Prophet - Goddess Arise**	**Steel Prophet - Goddess Arise**
17.	**Steel Prophet - Ghosts Once Past**	**Steel Prophet - Ghosts Once Past**
18.	Deicide - Behead The Prophet	Deicide - Behead The Prophet
19.	Deicide - Revocate The Agitator	Deicide - Revocate The Agitator
20.	**Steel Prophet - Dawn Of Man**	**Steel Prophet - Dawn Of Man**
21.	**Steel Prophet - 07-03-47**	**Steel Prophet - 07-03-47**
22.	Deicide - Holy Deception	**Steel Prophet - Mysteries Of Inquity**
23.	**Steel Prophet - Mysteries Of Inquity**	**Powergod - Metal Church**
24.	Deicide - Sacrificial Suicide	**Powergod - Burning the Witches**
25.	**Powergod - Madhouse**	**Iron Savior - Paradise**
26.	Crematory - Lost In Myself - Trance Raymix	**Powergod - Madhouse**
27.	Tiamat - Cain	**Powergod - Bleed for the gods**
28.	**Powergod - Bleed for the gods**	**Iron Savior - For The World (Live)**
29.	**Powergod - Ruler Of The Wasteland**	**Iron Savior - Brave New World**
30.	**Powergod - Burning the Witches**	**Iron Savior - Mindfeeder**
31.	**Powergod - Metal Church**	**Powergod - Stars**
32.	Crematory - Through My Soul	**Powergod - Ruler Of The Wasteland**
33.	Crematory - Reign Of Fear	**Powergod - Esper**
34.	**Powergod - Soldiers Under Command**	**Stratovarius - Rebel**
35.	Tiamat - Carry Your Cross An Ill Carry Mine	**Powergod - Soldiers Under Command**
36.	**Powergod - Stars**	**Iron Savior - Crazy (Ltd Ed Bonus**
37.	Crematory - Revolution	**Iron Savior - Iron Savior (Live)**
38.	Crematory - Red Sky	Electric Six - She's White
39.	Entombed - Left Hand Path (Outro)	**Powergod - Salvation**
40.	Monoide - One year after first love	**Powergod - Prisoner**
41.	**Finntroll - Ursvamp**	**Powergod - The Eagle & The Rainbow**
42.	**Finntroll - Grottans Barn**	**Powergod - Anybody Home**
43.	**Powergod - Esper**	**Powergod - Lost Illusions**
44.	**Iron Savior - For The World (Live)**	**Powergod - Tor With The Hammer**
45.	**Finntroll - Fiskarens Fiende**	**Iron Savior - Riding On Fire (Live)**
46.	**Finntroll - Nattfodd**	**Powergod - Red Rum**
47.	**Finntroll - Trollhammaren**	**Powergod - Steel The Light**
48.	Chicks on Speed - Procrastinator	**Iron Savior - No Heroes**
49.	Deicide - Crucifixation	**Powergod - I Am A Viking**
50.	Entombed - Say It In Slugs	**Powergod - Into The Battle**
51.	**Iron Savior - Mindfeeder**	**Powergod - Kill With Power**
52.	Crematory - Dreams	**Powergod - Mean Clean Fighting Machine**
53.	Tiamat - Light In Extension	**Powergod - Children Of Lost Horizons**
54.	Deicide - Mephistopheles	**Powergod - I'm On Fire**
55.	**Iron Savior - Brave New World**	**Powergod - Gods Of War**
56.	Tiamat - Nihil	**Powergod - No Brain No Pain**
57.	**Iron Savior - Paradise**	**Powergod - Observator**
58.	Crematory - Human Blood	**Powergod - Evilution Part I**
59.	Entombed - Something Out Of Nothing	**Powergod - Powergod**
60.	**Stratovarius - Rebel**	Corvus Corax - Bitte Bitte

Table 4. Results for the example query *speed metal*. Bold entries indicate relevant pieces according to the tags provided by Last.FM. Query update after 20 results.