Implementation of Relevance Feedback for Content-based Music Retrieval Based on User Prefences

Keiichiro Hoashi
KDDI R&D Laboratories, Inc.
2-1-15 Ohara Kamifukuoka
Saitama 356-8502 Japan
hoashi@kddilabs.jp

Erik Zeitler
Uppsala University
Signals & Systems, Box 528
SE-751 20 Uppsala Sweden
erik.zeitler@ieee.org

Naomi Inoue
KDDI R&D Laboratories, Inc.
2-1-15 Ohara Kamifukuoka
Saitama 356-8502 Japan
inoue@kddilabs.jp

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Storage and Retrieval—*relevance feedback*

General Terms

Algorithms, Experimentation.

1. INTRODUCTION

The main task of conventional music retrieval systems is to retrieve a music data, which matches the request of a user. The importance of such systems is expected to increase due to the rapid spread of digital music data formats such as MP3. However, conventional systems can only be used to search a particular song from a database. In this research, we investigate the performance of a conventional music retrieval method to retrieve songs based on the user's musical preferences. Furthermore, we propose the application of relevance feedback techniques to improve the music retrieval performance. Through evaluation experiments, we prove the effectiveness of our method.

2. TREE-STRUCTURED VECTOR QUANTIZATION

Foote has developed a tree-structured vector quantization algorithm (TreeQ)[2] for audio data, and has conducted music retrieval and categorization experiments based on this method. The approach of the TreeQ method is to train a vector quantizer (VQ) instead of modelling the sound data directly. Each audio datum in the training data set, which is a collection of audio data associated with a class such as artist or genre, is first parameterized into a spectral representation, by calculating mel-frequency cepstral coefficients (MFCC)[1]. Next, a learning algorithm constructs a quantization tree that attempts to put samples from different training classes into different bins (leaves). A histogram of an audio file can be generated by looking at the relative frequencies of samples in each quantization bin. The relative frequency can be considered as the probability of a data sample to end up in a certain leaf. If the resulting histograms are considered as vectors, typical vector similarity measures such as cosine similarity can be applied to calculate the similarity between any incoming audio data and category vectors. This method has been used for music and audio retrieval experiments. In Reference [2], experiments were conducted to to categorize music data based on musical genre.

3. PROBLEMS

If a sufficient amount of music preferences of a user is collected and used as training data for the TreeQ method, this approach can be applied to construct a system which retrieves music data based on user preferences. However, it is not realistic to expect a potential user of a music retrieval system to provide preferences of hundreds of songs prior to using the system.

Furthermore, the acoustical differences between songs which a user likes or dislikes is expected to be more ambiguous than the difference between songs of different genres, or between songs composed by different artists. Therefore, when the TreeQ method is applied on a small set of user preference data, the performance of the system is expected to be highly dependent on the selection of the songs within the training data set.

4. IMPLEMENTATION OF RELEVANCE FEEDBACK

Due to previously described problems, it is unlikely that a music retrieval system based on existing methods will achieve satisfactory performance. Therefore, in order to improve retrieval performance, we propose the implementation of relevance feedback. The outline of this method is illustrated in Figure 1.

In our proposed method, relevance feedback is applied to refine category vectors generated by the TreeQ method. This is accomplished by inputting relevant data through the VQ tree, and accumulating the results on the original category vector. The refined vector is used to retrieve a new list of music data. The approach taken here is similar to Rocchio's algorithm[4], where weights of terms occurring in relevant documents are accumulated to the initial query.

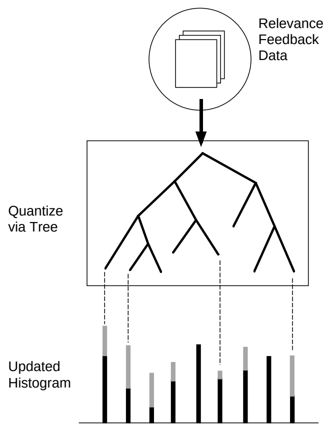

Figure 1: Outline of relevance feedback method

5. EXPERIMENTS

We conducted experiments to evaluate the effectiveness of relevance feedback for music retrieval. For our experiments, we collected user ratings of songs in a music data collection which consists of 107 popular songs composed by various artists. For each song in the collection, a rating ranging from 1 to 5 (Bad:1 $\sim$ Good:5) was given by experiment subjects. The rated data was then classified into 3 categories according to the subject ratings: the category of "good" songs (C_g), "bad" songs (C_b), and "fair" songs (C_f). Categories C_g, C_b, and C_f consist of songs rated (4 or 5), (1 or 2), and 3, respectively.

N songs were randomly extracted from each category, and used as training data for the TreeQ method. As a result of training, a vector which expresses each category is obtained. Vectors for test music data can also be obtained by inputting the data through the VQ tree. By calculating vector similarity between $\vec{C_g}$ and a test data $\vec{K}$, each song in the test data set can be ranked by similarity to category C_g ($Sim(\vec{C_g}, \vec{K})$), assuming that higher similarity to category C_g expresses higher probability that the subject will like the regarded song. Another scoring measure can be defined by the formula: $Sim(\vec{C_g}, \vec{K}) - Sim(\vec{C_b}, \vec{K})$, where the difference between the similarity of each test song and categories C_g and C_b is calculated. These two scoring measures are referred to as Sim (similarity) and $DiffSim$ (difference of similarity), respectively.

For relevance feedback, the top M-ranked relevant songs, i.e., songs which are included in C_g are extracted. The extracted songs are used to refine $\vec{C_g}$ by the method described in the previous section. $\vec{C_b}$ can also be refined by extracting the top M-ranked non-relevant songs (songs which are included in C_b) from the ranked list.

6. RESULTS

For evaluation of experiment results, the average precision is calculated based on each ranked list obtained per experiment. Results obtained based on the refined vectors are compared to the results prior to relevance feedback, which are referred to as the "baseline" results.

Due to the randomness of training data, the average precision of baseline results have a wide variance. Therefore, baseline experiments were conducted 10 times for each N, and relevance feedback experiments were conducted based on each baseline run. Furthermore, the mean improvement rate of average precision for all experiments per parameter set of M and N was calculated for performance evaluation, since average precision after relevance feedback is highly dependent on the regarded baseline results. Tables 1 and 2 show the mean improvement rate of average precision for scoring methods *Sim* and *DiffSim*, respectively.

Table 1: Improvement rate (%) of avg precision (*Sim*)

	Amount of training data (N)				
M	1	3	5	7	9
1	14.78	2.29	4.37	0.84	1.71
3	27.72	14.45	19.00	14.84	12.46
5	36.55	22.54	31.91	28.10	22.84
7	41.20	32.19	43.69	39.48	34.78

Table 2: Improvement rate (%) of avg precision (*DiffSim*)

	Amount of training data (N)				
M	1	3	5	7	9
1	27.68	5.88	10.22	2.52	4.74
3	40.94	19.70	23.80	17.20	27.50
5	52.15	32.14	34.08	27.99	40.59
7	62.89	43.45	43.76	36.45	52.89

As clear from these results, implementation of relevance feedback has constantly improved average precision. Relevance feedback was especially effective in cases where the amount of training data was low. It is also apparent that *DiffSim* has overperformed compared to *Sim*, which shows that relevance feedback of non-relevant information was also effective.

7. CONCLUSION

In order to improve the performance of content-based music retrieval based on user preferences, we proposed the implementation of relevance feedback. Experiments conducted on user preference data showed that the proposed method improved overall retrieval performance. We plan to increase the number of songs in the music collection and conduct similar experiments in the near future.

Acknowledgments

This research used the TreeQ package[3] developed by Jonathan T. Foote.

8. REFERENCES

[1] David, Mermelstein: "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences", IEEE Trans. Acoustic, Speech, Singal Proc., ASSP-28(4), 1980.

[2] Foote: "Content-based retrieval of music and audio", Proceedings of SPIE, Vol 3229, pp 138-147, 1997.

[3] Foote: "The TreeQ Package", `ftp://svr-ftp.eng.cam.ac.uk/pub/comp.speech/tools/treeq1.3.tar.gz`

[4] Rocchio: "Relevance Feedback in Information Retrieval", in "The SMART Retrieval System – Experiments in Automatic Document Processing", Prentice Hall Inc., pp 313-323, 1971.