T-61.184 Speech Recognition and Language Modeling: From Theory to Practice Course Project MUREA - MUsic REcognition Application

Ville Turunen, Jaakko Väyrynen, Jukka Parviainen

Laboratory of Computer and Information Science
Helsinki University of Technology (TKK)
{vt,jjvayryn,parvi}@cis.hut.fi

Abstract

A music recognition application (MUREA) is implemented for identifying songs from a database of songs. The system trains a GMM model for each song in the database using MFCC features extracted from raw audio. The time-independent models represent the spectral properties of the songs. Songs are identified by comparing the likelihoods of the models. Pruning and clustering methods are used to accomplish both fast identification and low error rate. The identification of artist and genre is also considered.

1. Introduction

Music recognition has some commercial products as well as contains some academic research. A British company Shazam Entertainment has created a service where the end-user holds a mobile phone in front of the radio loudspeaker, and after 30 seconds the service sends a SMS containing information on the song. The music archive contains some 2,200,000 songs and they receive over one million calls a year.[6]

The models and principles used in the speech recognition field can be adopted to basic music recognition. The standard Mel-Frequency Cepstrum Coefficient (MFCC) features describe the acoustic contents of an audio signal. Simplifying the speech recognition paradigm by discarding all time information, songs can be identified by their acoustic space modelled by Gaussian Mixture Models (GMMs). We have built a system for identifying songs based on the artist and title.

This paper is constructed as follows. First, the problem statement is described. Second, related literature is viewed for existing music recognition systems, as well as for the algorithms used in our approach. Third, our solution to the problem is described in detail with information about needed software. Next, the data and the results of our system are discussed. Finally, the project goals, future improvements and work distribution are discussed.

2. Problem Statement

A musical composition consists of one or more sources emitting notes in a sequential order. The sequence of notes with different frequencies and lengths can be used to define the composed song. The sources can be, for instance, humans singing, acoustic instruments plucked or hit, or natural or artificial sounds reproduced electronically.

Each source emits frequencies typical to its characteristics, with some variability given the different notes. Combined, the sources create a frequency distribution that changes over time. Assuming time-independence, each song can be described as a frequency distribution. Using this simplified but compact description, a song can be identified using a database of all songs.

3. Background Literature

Our project originates from the ideas in the course T-61.184 Speech Recognition and Language Modeling. Music recognition is considered to be similar to speech recognition, and basically the same tools are used.

In the automatic speech recognition (ASR) linear prediction coefficients (LPC), LPC-derived cepstral coefficients (LPCC), and mel-frequency cepstral coefficients (MFCC) are often used as features. MFCCs have been found most feasible in speech recognition. When studied with music samples, MFCCs were still the most suitable as well as the use of GMM was ensured [5]. GMMs are used to approximate probability density function of the observation samples (MFCCs). In Figure1 there is an example of 1-dimensional GMM with five Gaussians and 15 parameters totally. In speech recognition the dimension is typically 39 and the number of Gaussians $M = 5 \ldots 20$, which gives M weights, $39M$ means, and $39M$ variances, when using the diagonal covariance matrix.

There are also other possibilities to the identification like directly clustering the feature vectors. A self-organizing map (SOM) based system was used in [7].

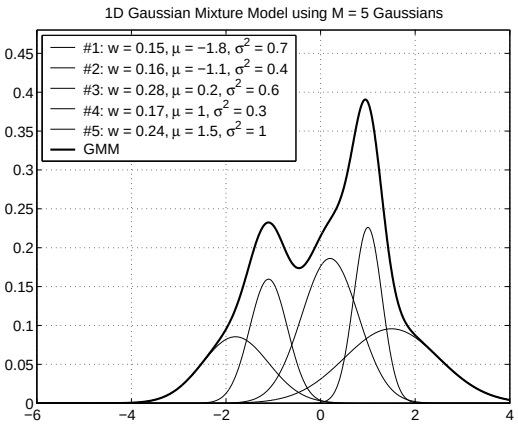

Figure 1: Example of 1-dimensional Gaussian mixture model (GMM) with 5 Gaussians.

A collection of 230 songs was analyzed using 484-dimensional feature vectors and mapped on a two-dimensional grid. This work was continued with extended features and a graphical user interface of "islands of music" [8].

Computing the distance between two GMMs was considered in [9]. The distance between two GMM density functions A and B is computed using "sampling method"

$$D(A,B) = \sum_i^T \log P(S_i^A|A) + \sum_i^T \log P(S_i^B|B) - \sum_i^T \log P(S_i^A|B) - \sum_i^T \log P(S_i^B|A)$$

where $P(S_i^X|Y)$ is the likelihood for observations sampled from model X given model Y. The parameter T (DSR, Distance Sample Rate) was evaluated and found to be 1000 for (8 MFCC's and 3 Gaussians) [10].

Given a distance matrix, samples can be clustered via agglomerative hierarchical clustering. Each sample is a cluster in the beginning. Clusters with minimum distance are merged together. Distance matrix is then updated using the maximum distance pairwise in the clusters. The size of the matrix decreased by one in each round. This is continued as long as there is only one cluster with all samples, or a priori number of clusters is reached. An example is shown in Figure 2. In the first step the clusters B and C are merged to a new cluster F. The distance matrix is modified by removing values shown with dashed line.

4. Objectives

Our objective is to build a working song identification software system. The users should be able to train their own databases and share them with others.

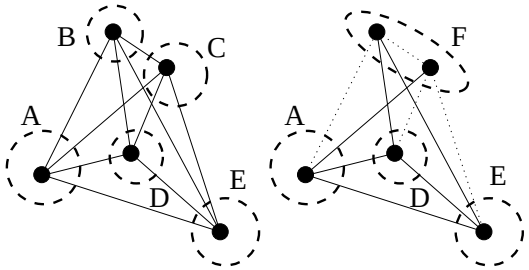

Figure 2: Agglomerative clustering is a bottom-up method, where all samples are initialized to clusters. The first step for five samples is shown in the figure.

4.1. Song identification

The system should be able to correctly identify most of the songs it has been trained on. Some identification accuracy may be sacrificed to achieve fast identification. At first, we will only consider the popular MP3-format as song input.

4.2. Artist and genre identification

In addition to song identification, we will attempt artist and genre identification. We will consider clustering song models and creating separate models for artists and genre. For clustering, the k-means algorithm is one possible choice.

4.3. Other features

The system should have an updatable database, to which new songs can be added. Adding a few new songs should not require retraining of the whole database.

4.4. Experiments

We will try to build a fast identification system. A possible enhancement to brute force search of all songs in the database is a heuristic search that considers only a portion of the songs in the database. The heuristic could employ song clustering.

5. System Design

Our system uses Gaussian Mixture Models for identifying songs. An overview of our system is given in Figure 3. A GMM is build for each song (and also for each artist). Training the GMMs involves a number of steps. First the MP3 file is converted to wav audio. Then the audio is converted to a series of feature vectors and a GMM is trained on these vectors. Identifying an unknown song involves testing the features of the song against the trained GMMs (acoustic models) and selecting the GMM which most likely produced the features.

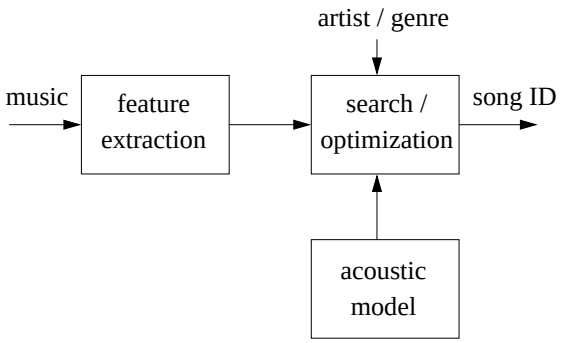

Figure 3: Overview on MUREA.

5.1. Data preprocessing

We used a set of 4000 MP3 files from 206 different artists to test the system. Our system ran on a 2800 MHz Pentium 4 computer. The main part of the system was written in C and some Perl-scripts were also written. We used some freely available software to help. To convert MP3 files to wav format, we used mpg123 [1] . Features were calculated using HTK toolkit [2]. Mp3info [3] was used to extract artist and genre information from MP3 files.

5.2. Feature extraction

A perl script was created to handle extracting features from all MP3 files. First, the MP3 file was decoded to wav format using mpg123. The audio was converted to mono and down sampled to 16 kHz sampling frequency. Then MFCC features were calculated from the wav-file using HCopy program from HTK toolkit.

MFCC features with 12 cepstrum coefficients, an energy measure, deltas and accelerations were calculated. We used 25 ms Hamming windows and 100 features per second were extracted. Later we noticed that the energy measure and its deltas get huge values at some points. Normalization would have helped, but to avoid calculating all the features again we decided to discard the energy measure and its deltas. So currently the system has been tested using 36 dimensional feature vectors.

5.3. Artist and genre information

MP3 files have id3-tags which have information about the artist and genre of the song. To build GMMs for each artist we needed a list of songs that belong to that artist. We wrote a perl script which generates this list using MP3info.

Similar models could also be built based on genre information of the songs. This would mean building a model for each genre. However, many files did not have genre information at all and often the genre information was not correct. Therefore, we decided not to build genre models.

5.4. Training Gaussian Mixture Models

The standard EM-algorithm [4] for training GMMs was implemented both in the linear domain and in the log domain. One GMM with a fixed number of Gaussians was estimated for each song to model the acoustic content.

The model parameters could be initialized either with random samples or using the mean and variance of all the samples. With random samples, mixture weights were set to be equal and sum to one, means were initialized with random samples and the covariance matrix was a unit matrix. In the latter case, initially there was only a single Gaussian and the learning proceeded by repeating splitting the Gaussian with the largest weight and learning the modified model until the number of Gaussians reached the required number.

To reduce numerical problems, the weights and variances of the Gaussians were checked after each parameter update. The weights were forced to be at least 0.01 by setting the value to the minimum value if it was below it. If a weight went below value 0.00001 the corresponding Gaussian was removed and the Gaussian with the largest weight was split into two.

5.5. Identifying songs

An unknown song is represented as a set of feature vectors. Identification of the song was accomplished by comparing the log-likelihoods of the song models for the input data and picking the model that gives the highest log-likelihood.

In order to speed up the recognition, nearest-neighbour approximation, where a single Gaussian is considered to be responsible for the whole log-likelihood, was implemented for the identification phase. Beam pruning and hierarchical search were also implemented to speed up the identification. Some speed gain over accuracy was achieved also by using only a subset of features in the learning and identification phases.

5.5.1. Brute force search

A brute force search linearly calculates the log-likelihood for each song model and selects the model with the highest value. The method is accurate and was considered as a base line method.

5.5.2. Beam pruning

Beam pruning was adopted from speech recognition to prune out unlikely models. The basic idea behind beam pruning is that by using some approximation of the complete log-likelihood, only models that seem promising at the moment are worth the computational cost of computing the final log-likelihood. The downside of reduced computational cost is that it might discard the right model.

We calculated an approximation of the log-likelihoods with a random subset of the features vectors. Models with log-likelihoods below a threshold were discarded and the log-likelihoods of the remaining models were re-calculated in the next pass using more feature vectors. The beam threshold was determined by subtracting a small fraction of the difference between the maximum and the minimum log-likelihood from the maximum log-likelihood. The fraction was determined by the "beam width" parameter. The beaming was repeated until there was only one model left or a predetermined number of iterations was reached.

5.5.3. Hierarchical search

In a hierarchical search, a set of models is approximated by a single model. We assigned one cluster for each song model and created GMMs for the clusters. To identify a song, the best N clusters were identified using a linear search over the cluster models, after which the search was repeated with the models in the selected clusters. The search hierarchy is illustrated in Fig. 4. The identification fails if the linear search fails either with the cluster models or the lower-level models.

We implemented a two-level hierarchical search using agglomerative hierarchical clustering, which is an unsupervised clustering method. Supervised clustering was also tried using the artist information for the songs.

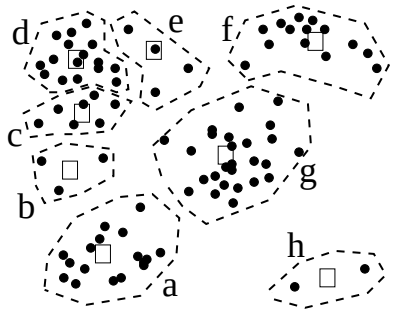

Figure 4: In a hierarchical search each song belongs to a cluster which has a prototype GMM (boxes). N best clusters are chosen first, and the linear search is continued only in those clusters.

6. Results

6.1. Brute force

To see how accurate the search can get, brute force linear search was tested. 4000 song models with 16 Gaussians each were taught. A random set of 750 songs was used for testing. At this test, all the features of the test song were used to calculate the likelihoods. Only four songs were misidentified giving the accuracy of 99.5%. However, the search was infeasibly slow. On average, identifying one song took 13 minutes. Nearest neighbor approximation was used. Because the accuracy was so good, we decided that it was not necessary to test the performance without the approximation.

6.2. Beam pruning

Beam pruning proved to be the most effective way to reduce search times without hurting the accuracy much. The same 4000 model database and 750 song test set were used. Three different beam widths were tested and the results are summarized in Table 1. At each iteration, the likelihoods of the remaining models were calculated by using 200 random samples. By further optimizing the search parameters such as beam width and number of samples used as well as model parameters, results could be improved.

	Beam width	Accuracy	Time
Brute force		99.5%	781 s
Beam pruning	0.1	99.3%	54.3 s
	0.01	98.9%	33.8 s
	0.001	98.3%	17.8 s

Table 1: Accuracy and average running time per song for brute force and beam pruning search. 4000 song database.

6.3. Hierarchical search

Clustering is a very time consuming process. To save time, the methods have so far been tested only on a database with models for 1000 songs using 8 Gaussian mixtures. The 1000 songs were selected randomly from the 4000 song database. All 1000 songs were also used for testing.

Two cluster hierarchies were built, one using agglomerative hierarchical clustering and the other using artist information, one cluster for each artist. The artist prototype models were estimated by randomly sampling features from all the songs from that artist. The database had songs from 114 different artists, so 114 models were estimated in the supervised case. In the unsupervised case 50 clusters were build. First, a linear search is done over the cluster models. N best clusters were selected and the song models in those clusters were searched linearly. The results for different values of N are summarized in Table 2. As before, results could be improved by optimizing parameters such as number of clusters used. Also, beam search could be combined with the hierarchical search to make the searches faster.

6.4. Artist identification

To further investigate how well the artist of the song can be identified, another set of artist models were build. This time only those artists were included who have 8 or more

	N	Accuracy	Time
Unsupervised	1	25.6%	1.9 s
	3	47.1%	3.1 s
	5	57.7%	4.4 s
	10	77.0%	8.4 s
Supervised	1	46.0%	3.3 s
	3	62.0%	3.7 s
	5	69.1%	4.5 s
	10	78.5%	6.3 s

Table 2: Accuracy and average running time per song for hierarchical search. 1000 song database.

songs in the database. For each artist, 75% of that artist's songs were included in the training set and the rest 25% were held out for testing how well the system can determine the artist of a song it has never seen.

Total of 2883 songs from 87 artists were included in the training. 87 GMMs with 16 Gaussians each were estimated. The test set consisted of 946 songs. The performance of the system was measured both with the songs in the training set and songs only in the test set. The identification was considered a hit if the correct artist was within the N best classification results. The results for different values of N are summarized in Table 3.

N	Training set	Test set
1	42.5%	30.7%
3	59.9%	46.7%
5	68.3%	57.0%
10	78.4%	71.4%

Table 3: Accuracy of artist identification.

7. Discussion

7.1. System performance

Our objectives for identifying songs were accomplished. Beam pruning is an accurate and fairly quick method for searching songs.

So far the system does not have an easy to use interface for creating and updating the database. Objectives concerning these features were not met. For beam pruning (and brute force) search, adding a new model to the database would be fairly easy, but for hierarchical search some or all of the cluster prototype models would have to be retrained.

The agglomerative hierarchical clustering did not work as well as expected. The supervised clustering using artist models works better, which suggests that the unsupervised clustering is done in an unoptimal way.

Models for identifying artists work as well as could be expected. The error rate is still quite high, but trying to model all songs from one artist by a single model is a hard task since songs by the same artist can vary quite a lot. Out-of-set songs were also classified quite well, only 10% or so less accurate than songs that were used in the estimation.

7.2. Future improvements

Throughout the process, there are many parameters that have not been optimized. In feature extraction, the parameters used were the same that typically used in speech recognition. For music identification these may not be very optimal. The estimation process could be improved by adjusting the number of Gaussians used and the number of features used. Better models could also be achieved by using incremental mixture splitting which was implemented but not tested.

The speed and accuracy of the beam pruning search could be improved by adjusting the beam width, number of samples used at each iteration and the maximum number of iterations. For hierarchical search, we believe that finding more optimal values for the number of clusters and the number of random samples used when estimating the prototype models could improve the results greatly. Another way of improving the hierarchical search would be adding more levels to the hierarchy.

7.3. Work distribution

There were at least three main problems in this project. First, a comprehensive archive of MP3 files had to be collected and processed in order to complete feature extraction. Second, an effective implementation of the training algorithm had to be written. Third, non-exhaustive search with a suitable clustering algorithm had to be executed. Responsible members for these three aspects were Ville Turunen, Jaakko Väyrynen, and Jukka Parviainen, respectively. Most credits of the successful project belong to Ville Turunen and Jaakko Väyrynen.

8. References

[1] M. Hipp, `http://www.mpg123.de/`

[2] The Hidden Markov Model Toolkit (HTK), `http://htk.eng.cam.ac.uk/`

[3] R. Cerqueira, C. Teff, `http://www.ibiblio.org/mp3info/`

[4] D. Reynolds, R. Rose, "Robust Text-Independent Speaker Identification Using Gaussian Mixture Speaker Models", IEEE ASSP, Vol. 3, No. 1, pages 72-83, January 1995.

[5] J. Marques, P. J. Moreno, "A Study of Musical Instrument Classification Using Gaussian Mixture Models and Support Vector Machines", Technical Report Series CRL 99/4, Cambridge Research Laboratory, June 1999

[6] Shazam Entertainment Ltd., http://www.shazamentertainment.com/

[7] M. Frühwirth, A. Rauber, "Self-Organizing Maps for Content-Based Music Clustering", Proceedings, 12th Italien Workshop on Neural Nets (WIRN01), Italy, May 2001

[8] E. Pampalk, "Islands of Music - Analysis, Organization, and Visualization of Music Archives", Diplomarbeit, Technical University of Vienna, December 2001, http://www.ai.univie.ac.at/~elias/music/

[9] J. Aucouturier, F. Pachet, "Music Similarity Measures: What's the Use?", Sony Computer Science Lab., Paris, France. IRCAM 2002.

[10] J. Aucouturier, F. Pachet, "Improving Timbre Similarity: How high's the sky?", http://journal.speech.cs.cmu.edu/articles/2004/3

A. Appendix

A.1. Usage of MUREA software

The quick guide for using MUREA

```
murea -h
```

Creating a new GMM song model database `db.gmm`

```
murea create FILENAME
```

Identification of a song using hierarchical search / beam pruning / linear brute force search

```
murea -c search FILENAME
murea -b search FILENAME
murea -l search FILENAME
```

Identification of a set of songs using hierarchical search

```
murea -c batch-search FILENAME
```