COMPARISONG: AUDIO COMPARISON ENGINE

Georgi Dzhambazov

Abstract: *In this paper we present CompariSong - a resolver of the query by humming task for a real audio information system. It compares a database of popular songs with a hummed-by-user melody and returns the closest matches. The proposed system performs audio indexing based on perceptual characteristics of the songs rather than the hitherto applied approach that uses manual metadata like text tags. We extract low-level audio features and transform them into a tangible indexing format. Then searching mechanism is applied that is based on comparison between the shape of the time series of extracted features. CompariSong is a step towards intelligent audio analysis as it uses a comparison criteria derived from natural human hearing.*

Key words: *audio indexing, Query by Humming (QBH), audio features extraction, feature transformation, data mining*

ACM Classification Keywords: *H.5.5 Information Systems - Information Interfaces and Presentation - Sound and Music Computing*

Conference: The paper is selected from Second International Conference "Intelligent Information and Engineering Systems" INFOS 2009, Varna, Bulgaria, June-July 2009

Introduction

Today digital audio information systems play a key role. Current information retrieval in the field is largely based on manually entered textual metadata like artist, title or style and relies on search techniques like document indexing. However these are not able to reflect the abstract perceptual information present in the audio signal itself.

This work presents an engine Compari*Song* that addresses the query by humming (QBH) problem. It is the scenario in which a user hums a query melody and searches for a closest match in a music database. The System compares the hummed melody line with the main vocal line of each target piece. The comparison criteria are based on the shape of time series of selected audio characteristics. These are first extracted from audio entries and then analyzed and transformed in a tangible for comparison format.

Our database consists of audio files that are vocal-only counterparts to real polyphonic popular songs, e.g. melodies with no background instruments.

The paper is organized as follows: we first describe what the components of the system are. Afterwards each component is described in detail. Then implementation considerations are outlined followed by analysis of related work. In the end we summarize the importance of the proposed system and point out some future work.

System Decomposition

The proposed system Compari*Song* consists of three logical parts that are depicted in figure 1. : Preprocessing, Indexing and Comparison. Preprocessing is concerned with extracting the main voice party from each polyphonic recording from the initial music database. The Indexing part is essentially analysis of the characteristics of the audio entries and preparing them for comparison. Indexing itself consists of two parts: Feature extraction and Quantization. The indexing phase is applied first to each song from the database and then to the query. The

Comparison part conducts the search. It first calculates similarities between each entry from the database and the query and then ranks them.

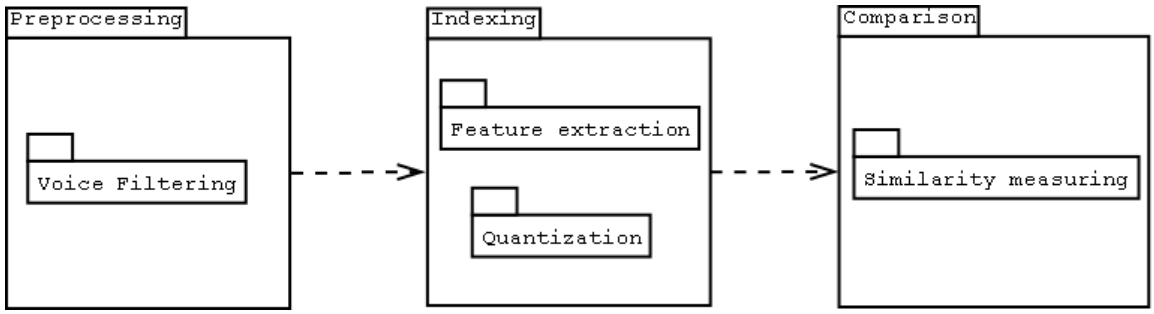

Figure 1. System Components

Preprocessing

Voice Filtering

Typical users querying the system would hum patterns from vocal melody, but not instrumental melodies. Therefore we are interested in extracting the leading voice part from database entries and then working only with it further. This is needed because in polyphonic audio multiple instrumental parts are mixed into a one-track signal which dissolves information for the separate instruments and voices. Filtering leading melody directly from pop music is prone to errors. Melody line is normally outlined by a dominating over the background instruments lead singer. However, on feature extraction simply taking the tones with maximum feature value does not exclude random notes from rhythm or back-vocals with high intensity that are weaving in the melody line.

The current work does not implement a vocal filtering phase. Its application would result in unreliable results since for this problem there is no universally effective algorithm devised so far. Instead we construct manually a database of vocal-only counterparts to the real polyphonic popular songs in the original database.

Indexing

Feature Extraction

Audio features are typically divided to low-level and high-level. We will make use of low-level audio features: once extracted from the audio signal, a low-level feature is ready to be used without any further processing. Digital audio signal is a function of time and consists of sequence of samples [5 Steiglitz]. In order to retrieve genuine information about some audio descriptor we use an extraction method that analyzes the signal on a per-window basis. The windows are equal-sized intervals of samples of length less than a second. They are the smallest logical audio units. Any audio feature is a parameter defined for a given time window and therefore is also a function of time. After feature extraction we arrive at time series of scalar feature values with length equal to the number of the windows in the analyzed song.

A judgement for the size of the window used could be argued. We adopt a 1/8 of a second window to reflect the most common minimal musical unit - namely the length of a sixteenth note in moderate (120 bpm) tempo.

The following audio descriptors are taken into account in the current work.

Melody contour

Each audio signal has a wide spectrum of frequencies at a given moment. Only one of them however is important for the audible pitch of the signal. This is the frequency of the tone with strongest audio power. Althogh not always the case, it usually determines the pitch of the sounding tone. Therefore a strongest frequency is taken as a good estimate for pitch of the tone. The sequence of the pitch characteristic defines the contour of the melody

line. Figure 2 presents an example of a melody contour. Tones in the melody are presented by dark parallelograms. On the vertical axis is pitch and on the horizontal time. A wider parallelogram means a longer tone. A higher one indicates higher pitch.

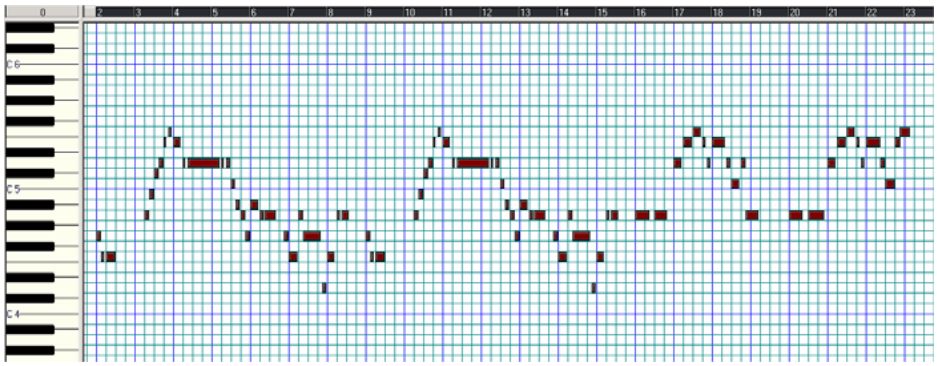

Figure 2. Melody Contour for *Yesterday* by *Beatles*

Audio Power Contour

Relative Audio power describes the average intensity of the signal for a given time window. The varying intensity and the pitch of a sung melody are closely related. Accents normally emphasize specific melody parts and outline patterns. The contribution of relative power is undisputable due to the fact that even if users sing false they tend to regenerate the accent sequence within a melody.

Quantization

After extracting features from the signal we arrive at feature values in the form of raw numeric time series. A quantization to a discrete alphabet transforms the data into sequence of symbols {A, B, …} and is proposed by [6 Stober]. Quantization consists in defining a lookup table that presents a set of bin values that divide the range of a given feature distribution into equally–sized intervals. Such a table is presented in figure 3. Important parameter is the size of the alphabet and the corresponding interval width. This is adjusted empirically for each feature.

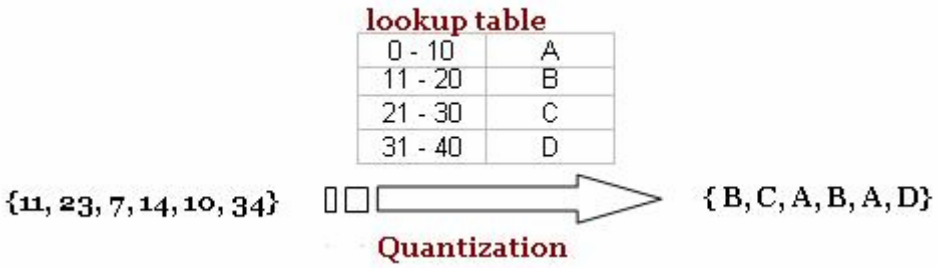

Figure 3. Quantization method

We introduce a fragmentation step at which the time series of feature values of a target song is divided into fragments with size equal to the size of the query. Then discretization takes place locally for a fragment, but not for the target as a whole. A local lookup table is constructed for each fragment individually. This includes taking the local minimum value of the feature distribution as a starting bin value. Working in fragments slows down the algorithm but is more precise. Additionally, it assures that a bias caused by a global minimal value that is an outlier would be avoided.

The symbolic representation allows us to use a well-known similalrity measure algorithm. Other advantage of the string format is that it reveals common patterns and shapes and makes the time series more legible [6 Stober].

Furthermore, discretization to a common alphabet normalizes time series of target piece and query with respect to absolute value of the series values.

Comparison

Similarity measuring

In this part the system compares the shape of the time series of each feature. Our engine aims at determining the target song with closest shape of the observed audio descriptors. Comparison is essentially the process of measuring the similarity between the string fingerprints. Figure 4 illustrates the shape of the parameter audio power for a target and a query. On the horizontal axis are the time samples and on the vertical is the value of the feature. The third diagram shows how the query is "slid" through the target until finding the best match.

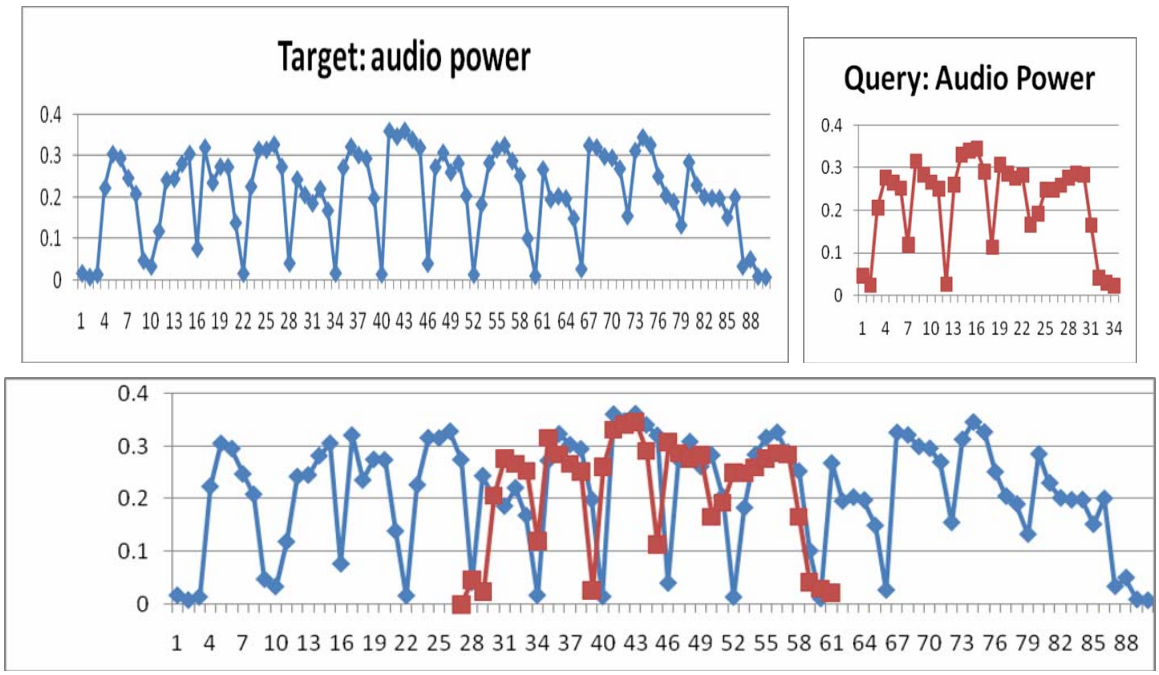

Figure 4. Comparison of shape of time series for the audio power feature

For a target song each audio descriptor is compared individually. For each descriptor the same common distance metric is defined – it computes a numeric distance from the query to a target song.

We apply a metric called *shift edit distance* that is based on the proposed by [1 Duda] *continuous edit distance.* It is in turn based on the Levenstein distance metric which calculates the number of operations needed to alter a string - symbol by symbol - so that in the end it equals another one [7]. The more operations, the more different the strings are. Each alteration operation has a cost weight assigned, which is in theory a fixed discrete number independent on the specific string symbols involved in the operation.

Continuous edit distance introduces non-discrete costs in order to tolerate melody tones with different but close characteristics. We apply the following formula (1) for a replacement operation. Here X and Y are some symbols of the alphabet and cost(X,Y) is the cost of replacing a symbol X with Y

$$\text{cost}(X,Y) = \frac{dist(X,Y)}{size\ of\ \ alphabet}; \quad dist(X,Y) = \left|index(X) - index(Y)\right| \qquad (1)$$

Before any comparison is conducted the target is divided into fragments as explained in the Quantization part. The shift edit distance takes place in windows starting at the first fragment and shifting at one position to the right so that the pieces are overlapping. This way all possible parts of the target song are compared each in turn with

the query. Figure 5 illustrates that process with target of length 8 symbols and query of length 5 symbols. Here 3 shifts and 4 comparisons are needed. Each square represents a window.

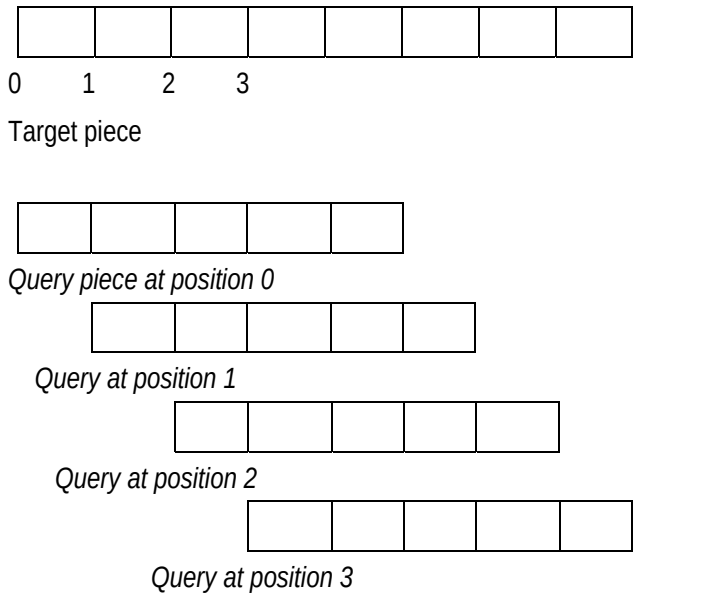

Figure 5: Shift Edit Distance

The distance of the query to a fragment is the sum of all letter-by-letter costs. As final feature distance only the recorded minimal fragment distance is considered as it is the closest match for the target song.

In order to define an overall feature distance we need to merge all feature distances into a quantitative metric that is defined by the formula:

$$\text{overall distance} = \sum(\text{weight}*\text{feature distance}); \quad \sum \text{weights}=1 \tag{2}$$

Weights are assigned in order to emphasize some features. For the current work empirical weight adjustment showed that the pitch is equally deterministic measure to the power level, which led to two equal weights of 0.5. The final step comparison engine commences is ranking of the results according their assigned distance.

Implementation

An implemented version of the current work was developed using java programming language. It is a desktop application with simple and intuitive GUI using the library Standard Widget Toolkit (SWT).

For extracting the features a reliable descriptor analyzer tool called jaudio [8] is applied. For the current implementation we rest on the „fundamental frequency via FFT maximum“ as pitch estimator and the „Root Mean Square“ (RMS) as an audio power estimator.

Related work

QBH research was first successfully conducted for databases that consist of MIDI entries in [2 Ghias]. However as MIDI lacks the perceptual completeness of real-performance recordings they are less intuitive to a user.

In [6 Stober] and [3 Hung Ming] a QBH system for polyphonic melody is presented. He proposes quantization based on the distribution of feature values for each song. This ensures that the alphabet is optimally utilized for

each song. We however, take even more precise approach by taking into consideration fragments of the target song.

In [4 Lin] a thorough hierarchy of different approaches for representation of time series of a given parameter is listed.

Conclusion

In the current work we presented Compari*Song*: an audio comparison engine. it is a step towards intelligent audio analysis as it uses a comparison criteria perceptual audio characteristics typical for natural human hearing.

Although it works with sung target database, that is manually annotated, we deem that a robust melody extraction method could be easily integrated, which will allow the application of the engine to real audio databases directly. This could be achieved for example through a vocal filter. Additionally, a web service could be integrated in order to discover and load an exhaustive on-line target music collection. These would make the system practically more applicable.

Bibliography

[1 Duda] Alexander Duda: Query by singing/humming with low-level feature extraction.

[2 Ghias] Asif Ghias: Query By Humming - Musical Information Retrieval in an Audio Database

[3 Hung Ming] Yu Hung-Ming : A Query-by-singing Technique for Retrieving Polyphonic Objects of Popular Music

[4 Lin] Jessica Lin: A Symbolic Representation of Time Series, with Implications for Streaming Algorithms

[5 Steiglitz] Ken Steiglitz: A Digital Signal Processing Primer

[6 Stober] Sebastian Stober: Towards query by singing/humming on audio databases

[7] Levenstein distance http://en.wikipedia.org/wiki/Levenshtein_distance

[8] jaudo http://jaudio.sourceforge.net

Authors' Information

Georgi Dzhambazov –American University in Bulgaria, e-mail: gbd040@aubg.bg