AudioSearch

# Introduction

An AudioSearch application, with its dataset consists of environmental sounds of indoor/outdoor scene. A combination of four Audio matching algorithms – namely Magnitude Spectrum, Energy, Zero-Crossing, and Mel-Frequency Cepstral Coefficient (MFCC) – were implemented to compute the similarities between a queried audio clip and audio clips within the database.

In this report, the approach in implementing the AudioSearch will be discussed. It covers various methods including (1) capturing relevant information from the training data, (2) calculating relevant probabilities or scores, and (3) combining the audio matching algorithms together.

# Implementation Details

## Training Data

The training data used by the program contains environmental sounds of 10 indoor/outdoor scenes, such as bus and office, which are provided with the ground truth, categorizing the data into at least one of the 10 classes.

Please refer to the training data summary below:

|  |  |
| --- | --- |
| **Category** | **Count** |
| Number of training audio clips | 1250 |
| Number of test audio clips | 100 |
| Number of unique category | 10 |

Table 1 – Summary of training data

## Information Extraction

For each audio clip in the training database, the following information is extracted:

|  |  |
| --- | --- |
| **Item** | **Description** |
| Magnitude Spectrum | An array (vector) of floating point numbers |
| Energy | An array (vector) of floating point numbers |
| Zero Crossing | An array (vector) of floating point numbers |
| MFCC | An array (vector) of floating point numbers |
|  |  |

Table 2 – Information extracted from training data

## Pre-Processing of Audio clip Database

The training data forms the program’s audio database. It was noticed that extracting information (the feature vector) from every category (Magnitude Spectrum, Energy, Zero-Crossing, and MFCC) for each audio clip took a considerable amount of time (up to 5 seconds per audio clip). Thus to do so for all 150 audio clips in an on-demand manner (i.e. when a query audio clip was given) is not practical.

Hence it was decided that the information for each audio clip in the databases were to be pre-processed, and stored in a custom database (similar to an index). Thus when running a query against the database, the information is taken from this custom database (or index) instead of directly from the audio clips itself.

## Similarity Measure

Three similarity measures will be used in AudioSearch namely Cosine, Euclidean, and City-block distance.

Cosine similarity for is defined below:

Euclidean distance is defined below:

City-block distance is defined below:

## Magnitude Spectrum

The audio clip is first processed and read, producing an array (vector) of integers. The input signals are then converted into frames. For each frame, Fast Fourier Transform (FFT) is used to obtain the value of Magnitude spectrum.

## Energy

The audio clip is processed in the same way as Magnitude Spectrum and converted into frames. Energy is the sum of energy of all samples in a frame.

## Zero-Crossing

Zero-Crossing rate is the rate of sign-change along a signal. Using the quantized/processed signal, zero-crossing rate can be computed resulting in a floating point number.

## MFCC

MFCC is are the cofficients that collectively make up an MFC. Pre-emphasis is done to the input signal, and then Hamming window is applied to all frames. FFT with other complex computation is then performed to process each frame and producing a vector of floating point numbers.

## Overall Score

The final score used for each of the audio clip is the sum of 3 scores from using 3 different similarity measures against the 4 scores from the 4 features after normalization (0.0 to 1.0 range). Then each score is multiplied by some weight.

Score(i,j) denotes that score value is obtained by comparing feature “i” and using similarity measure “j”.

Based on the final score, the audio clips in the database are ranked, and the top 20 are shown to the user. The weights are obtained by running the system with different combinations and evaluating its Average Precision (will be discussed later). The best weights with the highest performance (Average Precision) are selected.

In this case, there are 7 unique weights (4 features weights, and 3 similarity measure weights), and all of the weights will be obtained using Genetic Algorithm described below.

## Genetic Algorithm

For the system to work well, an optimum weight for each feature is required. Manual prediction by a human hand might not result in the optimum weight sets. To overcome this problem, a learning algorithm is adopted, namely Genetic Algorithm. It is used to learn and find the optimum weights, given initial random weights and a training data set and Average Prevision is used to measure the performance of each weight set (the fitness of the gene).

The Genetic algorithm starts by reading a few specified set of weights defined by user. It creates offspring by permuting gene pairs, and crossover of each weight (chromosome) is performed on the two parents. Crossover cutting points are not fixed, and they are decided randomly on whether to pick the chromosome from the first parent or the second parent. Mutation method chosen in Genetic Algorithm is highly important in finding the optimum weights. Small steps are often successful, especially when the genes are already well adapted to the environment. However, large steps are also necessary to prevent the weights to be caught in local maxima. Larger changes can also produce good result much quicker when the genes are not well adapted yet. Thus higher probability small-steps mutation and lower probability large-steps mutation are adopted. The genes are sorted based on the best Average Precision, and 5 genes are brought forward to the next iteration.

Genetic Algorithm is chosen due to the following reasons:

* Its simplicity to implement within the program itself
* It is easy to recognize the level of significance of each feature

If other machine learning techniques are used, e.g. Maximum Entropy or Support Vector Machine, it is not possible to tell which feature contributes the most (and the least) to the performance of the AudioSearch system.

## Relevance Feedback

Simple Relevance Feedback was implemented. User can simply left click the audio clip to give positive relevance feedback (i.e. the audio clip is relevant), and right click the audio clip to indicate negative feedback (i.e. audio clip is irrelevant). It works by adding/subtracting a substantial amount (50%) of all the audio clips similarity score similar to the chosen audio clip.

# Evaluation

Mean Precision is used to evaluate the performance of AudioSearch because it is more relevant as compared to recall. For each query, recall and precision are defined below:

Relevant audio clips are defined as audio clip with one or more common category/class as the query audio clip.

Each of the 100 test audio clips is used to query the 1250 training audio clips. For each query, the Precision is calculated, and the average of the Precision for all queries is the performance measure of AudioSearch. The evaluation results, including each individual features, are as follows:

|  |  |  |
| --- | --- | --- |
| **Distance Measure** | **Features** | **Mean Precision @ 20** |
| Cosine | Magnitude Spectrum | 0.608000 |
| Cosine | Energy | 0.12249999 |
| Cosine | Zero Crossing | 0.099999 |
| Cosine | MFCC | 0.6220000 |
| Euclidean | Magnitude Spectrum | 0.58799999 |
| Euclidean | Energy | 0.25050000 |
| Euclidean | Zero Crossing | 0.19250000 |
| Euclidean | MFCC | 0.61550000 |
| City Block | Magnitude Spectrum | 0.58799999 |
| City Block | Energy | 0.25050000 |
| City Block | Zero Crossing | 0.192500000 |
| City Block | MFCC | 0.615500000 |
| **Combined with best weight** | **Combined with best weight** | **0.6605** |

Table 3 – Evaluation Results of AudioSearch

Some of the best results from Genetic Algorithm are as follows:

|  |  |
| --- | --- |
| Weight of Magnitude Spectrum | **W1** |
| Weight of Energy | **W2** |
| Weight of Zero Crossing | **W3** |
| Weight of MFCC | **W4** |
| Weight of Cosine | **W5** |
| Weight of Euclidean | **W6** |
| Weight of City-block | **W7** |

Table 4 – Feature Weight Mapping

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **W1** | **W2** | **W3** | **W4** | **W5** | **W6** | **W7** | **Mean Precision @ 20** |
| **462.608** | **545.31** | **786.846** | **362.750** | **994.886** | **288.432** | **-148.183** | **0.6605** |
| 875.933 | 580.451 | 343.333 | 579.705 | 474.464 | 310.436 | -186.921 | 0.6585 |
| 828.712 | 773.446 | 328.426 | 651.037 | 735.392 | 887.360 | 798.174 | 0.6580 |
| 580.525 | 759.909 | 471.586 | 536.540 | 875.276 | 648.686 | 531.664 | 0.6570 |
| 488.896 | 716.315 | 471.586 | 536.540 | 915.256 | 648.686 | 531.664 | 0.6540 |

Table 5 – Best Weights from Genetic Algorithm

# Conclusion

The best precision we could obtain has a mean precision score of 0.6605, and is a result of using all 7 features with their respective weights produced by the genetic algorithm.

When viewing the individual weights (W1 to W7) of the best mean precision score, we can conclude that all features – both the features for audio matching and features for similarity measures – are important to obtain good precision. When taken as a group with the respective weights, the overall precision is higher than any individual feature, thus using all features (with its respective weights) would be highly recommended.