Audio Segmentation using Support Vector Machines, Random Forest Classifiers, and K-Nearest Neighbors

Matt McBrien and Vishnu Tanguturi[a]

Georgia Institute of Technology[a]

1 Introduction

Audio Segmentation is a task primarily viewed as a preprocessing technique that greatly improves efficiency in further analysis of audio streams. In order to be able to segment audio effectively, there must be a methodology of classifying frames of an audio file into predetermined audio classes. Methods to classify audio frames basically require audio features to be extracted from which a classification model can be learned. Audio segmentation has been approached by the following methodologies: model-based, metric-based and energy-based segmentation as in [1]. We implemented a solution that is a combination of model and metric based segmentation. This solution has already been reached through various existing algorithms in the machine learning community.

Some approaches include using an E-M algorithm in Hidden Markov Model (HMM) [2], a methodology that focuses on temporal segmentation focusing on distance metrics between successive frames [3], a heuristic approach that was rule based, determined through characteristics of each audio class, and directly linked to features extracted [4], a clustering approach as in [5] and a Support Vector Machine (SVM) approach as approached in [6].

The approach using a heuristic method [4] was not favored because it was too procedural in the sense that a different algorithm and set of rules were necessary to handle each distinct audio class. In addition, the HMM model was not followed because this procedure was mainly to distinguish two very similar classes and required large amounts of data to optimize the E-M approach. Our reason for selecting the SVM classifier to handle this audio processing is to ensure that the distribution of audio data that occurs in a high dimensional feature space is well handled. Therefore, our approach follows the procedure in [6].

The ability to include this technology as a front-end to any audio processing is very applicable. Audio archive management, music information retrieval systems, audio surveillance systems and even video segmentation are all problems whose solutions would increase in efficiency with proper audio segmentation completed.

2 Problem Formulation

2.1 Project Overview

The goal of this project is to classify segments of audio into the following five audio classes:

- Pure Speech is defined as a single person speaking with zero to minimal background noise. In our dataset, only English speakers were used.
- Impure Speech is defined as an audio segment that includes a person speaking with any type of background noise. This could be gaussian white noise, instrumental music, or other people talking.
- Music is defined as any audio segment that contains instrumental music without a person talking or singing.
- Noise includes a wide range of audio signals from gaussian white noise to a group of people with indiscernible speech to a car being revved.
- Silence is defined as an audio signal below a certain energy level.

In order to solve a multi-class classification problem, we need to be able to determine a set of features that will accurately represent the raw audio data and be effective in distinguishing the characteristics for each audio class. From the aforementioned previous work, and topics discussed in class, we selected a base feature set of: energy, zero-crossing rate, mel-frequency cepstrum coefficients, spectral centroid (brightness), spectrum flux, bandwidth, pitched ratio. These features have high variance between the audio classes discussed above, so they are strong candidates as inputs to a supervised classifier.

We need to compare the results of the SVM against other types of classifiers to determine if any performance boost is possible against the traditional SVM approach. We chose to also use Random Forest and K Nearest-Neighbors (kNN) classifiers. The Random Forest Classifier is an extremely popular and versatile machine learning technique. Due to its popularity and off-the-shelf performance, if another algorithm outperforms it, then the outperforming algorithm is a candidate for further investigation. Due to the nature of using distance based measures to differentiate between different audio classes as in [3] we also wanted to test the ability of kNN classifier.

2.2 Design Approach

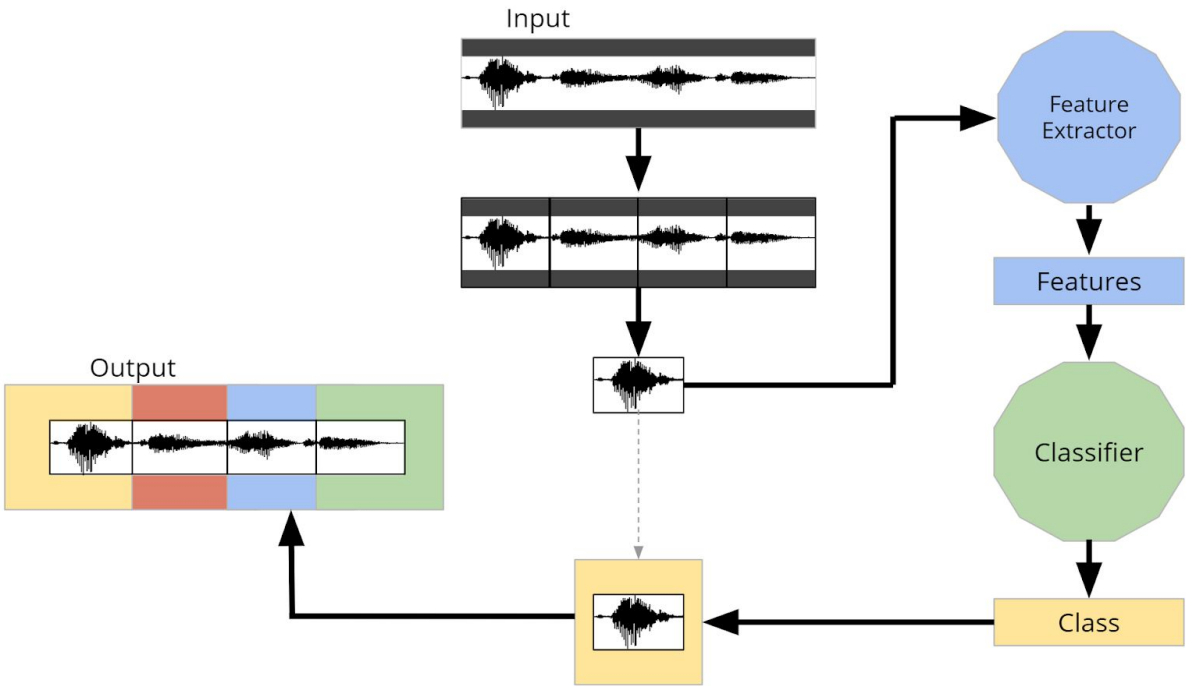

Figure 1. Audio Segmentation Approach

As can be seen in Figure 1, we followed an approach that first was able to represent the audio files by features and then applying classifiers on the feature vectors. The raw audio is divided into 1 second frames. These 1 second frames are then subdivided into forty 25 millisecond subrames. Then for each subframe we extract the set of features aforementioned and obtain the average and standard deviation for the frame. This way every 1 second of data is represented by a feature vector of length approximately 40. We build our dataset this way, with every 1 second of data being a separate data point that belongs to a certain audio class.

After training our classifier according to this methodology, we then validated our classifiers on the testing data which allowed us to see if the model needs to be tuned or additional features need to be added. Thus far, we were able to accurately classify 1 second frames of audio data into one of the five audio classes previously defined. Following this milestone, we proceeded to obtain another form of results; coming from testing against what we are calling "real world data." Our dataset collected for training and testing contained homogenous audio files, that is consisting of only one type of audio class. This ensured that our classifiers could be trained on appropriate, well-labeled data. However, in the real world, audio is jumbled together as can be seen in [7]. For example in movies, there is dialogue, music, explosions, and all sorts of sounds. The original motivation for this project was to be able to classify such audio signals on a second-to-second basis so that large audio clips can be automatically classified and rearranged into their distinct parts.

3 Experimental Configuration

With typical classifying problems a dataset must be assembled; the data must be processed for useful information; this processed data must be used to train a classifier; and then this classifier must be tested against unseen testing data. We took this approach with our own experiments to determine the accuracy of different supervised classifiers.

A dataset had to be assembled that contained a large set of audio from all classifications: pure speech, impure speech, music, noise, and silence. After collecting data we used our feature extractor to extract meaningful features from the audio (this is discussed in detail below). With the features extracted, we had meaningful input to feed into the classifiers which could then be fitted and be used for predictions.

3.1 Data Collection

Dealing with different classes of audio that was desired to classify audio into, it was hard to obtain relatively large amounts of data for each class. The primary source for our data came from datasets publicly available on Kaggle. We utilized the Urban Sound Classification dataset to obtain audio files for music and background noise. In addition, we used the GTZAN dataset (a well known dataset for its application in research to music genre identification) which had files that we obtained for impure speech and music audio classes. We also used some files within the TIMIT database to add more data for the pure speech and impure speech class. In order to obtain more data for background noise, we used OSU's noise database that was provided to us as part of the class. For the silence audio class, we generated gaussian noise and reduced the standard deviation.

As mentioned earlier, our code went through all the audio files that we obtained and split them into 1 second frames, and each second was a data point for our training and testing data. After training our model, and validating the model results on the testing data, we also downloaded multiple internet videos to apply our trained classifiers on "real-world" data.

3.2 Development of Feature Extractor

A custom feature extractor was built for this project. We extracted many of the same features found in [6] and some others that were not. Specifically, we extracted the following list of features from each clip.

1. Energy which was calculated by the sum of squares of the audio segment. This feature can identify silence alone as silence is defined by the energy level.
2. Zero crossing rate which is how often the value of the signal crosses zero divided by the length of the signal.
3. Spectral Centroid which is a measure that indicates where the "center of mass" of a spectrum is located. Perceptually, it follows with the "brightness" of a sound.

4. Spectral Flux which is a measure of how quickly the power spectrum of a signal changes. The comparison is often determined by distance functions like Euclidean distance between two frames. This is usually used to determine the timbre of an audio signal.
5. Bandwidth which is a frequency range (between 80Hz - 12kHz) including all harmonics that characterizes a normal speaker. This feature is critical to decide if audio is speech or not. This is perfect in the instance of using SVM as there is a one-vs-one approach.
6. Harmonic Ratio which is a comparison between the amount of energy within periodic speech cycle and within a noise cycle. Thus, it is a measure of voice quality. This feature is a good indicator of data belonging to the impure speech audio class.
7. Pitch which is a measure of the fundamental frequency that occurs in voiced speech. This allows different sounds to be organized.
8. Mel-frequency cepstrum coefficients which are a set of 12 coefficients that represent the envelope that is present in the short time power spectrum that represents the vocal tract. These features allow for the shape of that envelope to be determined, which gives an accurate representation of the phoneme being expelled. This feature is predominantly used for automatic speech recognition (ASR) but in the case of audio segmentation provides an easy way to determine cases of pure and impure speech.

For each of these features, the mean and standard deviation was calculated across the 40 25ms subframes within each 1 second frame.

3.3 Supervised Classifier Application

Once the data is formatted and processed, we used different supervised classifiers to make predictions on the data and learn how insightful the selected features are. There are many different types of supervised classifiers, and some have been used for this very problem. In [6], both support vector machines and the k-nearest neighbor algorithm were used. Both of these classifiers were implemented in our experiments along with the Random Forest Classifier.

The Random Forest Classifier is a very popular classifier. The Random Forest Classifier creates many decision trees from randomly sampled subsets of the dataset. Each decision tree attempts to split its subset into the cleanest divisions by finding point each feature that divides the classes. For prediction, the Random Forest sends the new data into each decision tree and selects the most popular prediction from the entire forest for the final prediction. Random Forests are popular for their straightforward implementation, their lack of “black box” behavior, and their high accuracy when making predictions.

After all three classifiers were trained, it was possible to compare their effectiveness in this problem space. Different classifiers tend to perform differently depending upon the type of classification problem, so by using all three classifier, it is possible to gain insight into how these perform in the audio segmentation problem.

We used the python package sklearn to implement the above three classifiers and tuned the hyper parameters to maximize the accuracy of each classifier.

3.4 Implementation Issues

The main implementation issues encountered in this project have to do with data collection and data processing. We were not able to find a specific dataset that contained audio from our five classes. This meant we had to bring together many disparate datasets that contained data from a single class. Additionally, collecting homogeneous data (i.e. collecting data that contained only one class and no others), was challenging.

4 Results

Our results come in two forms. As in typical classification problems, we split our data into train and test data for the purposes of evaluating the accuracy each classifier. A 80:20 train:test split was selected and each classifier produced an overall accuracy and a confusion matrix that can be analyzed.

Table 1: Accuracy for each classifier

	Support Vector Classifier	K-Nearest Neighbor Classifier	Random Forest Classifier
Accuracy (%)	94	83.1	91.8

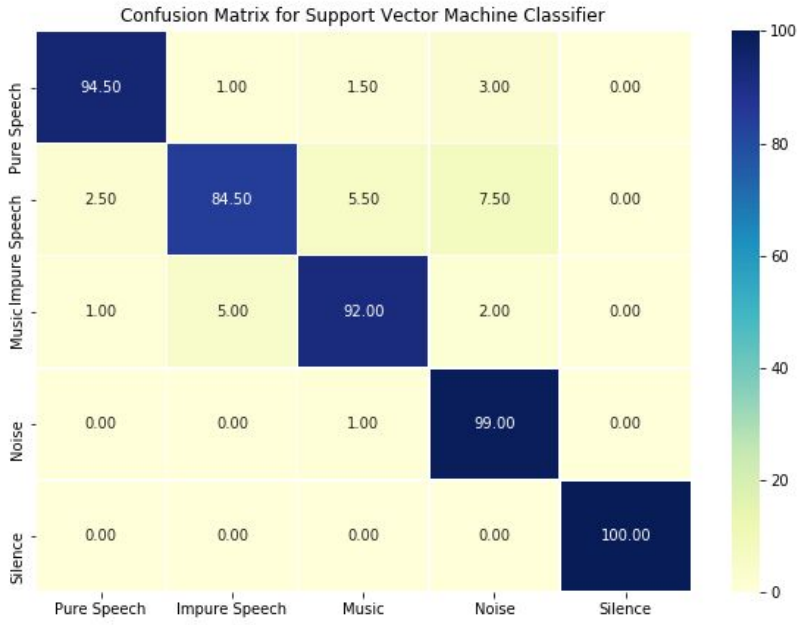

Figure 2. SVM Confusion Matrix

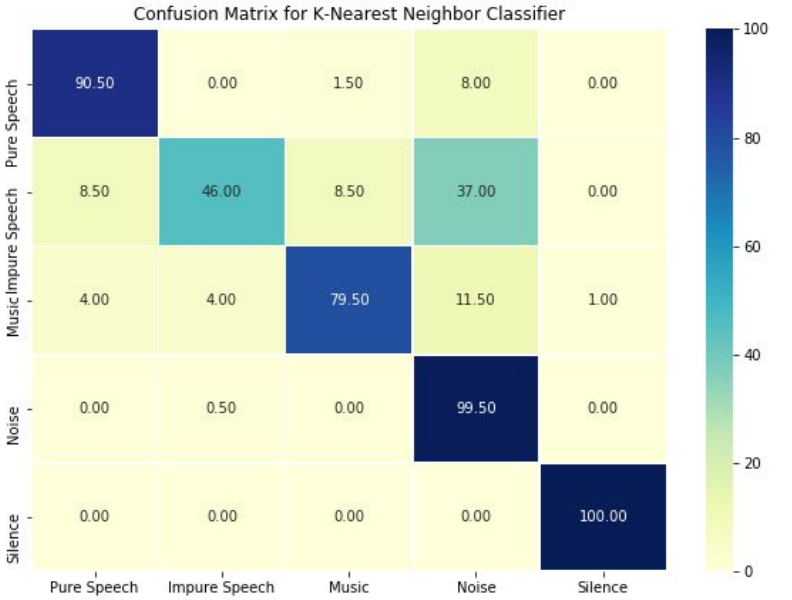

Figure 3. KNN Confusion Matrix

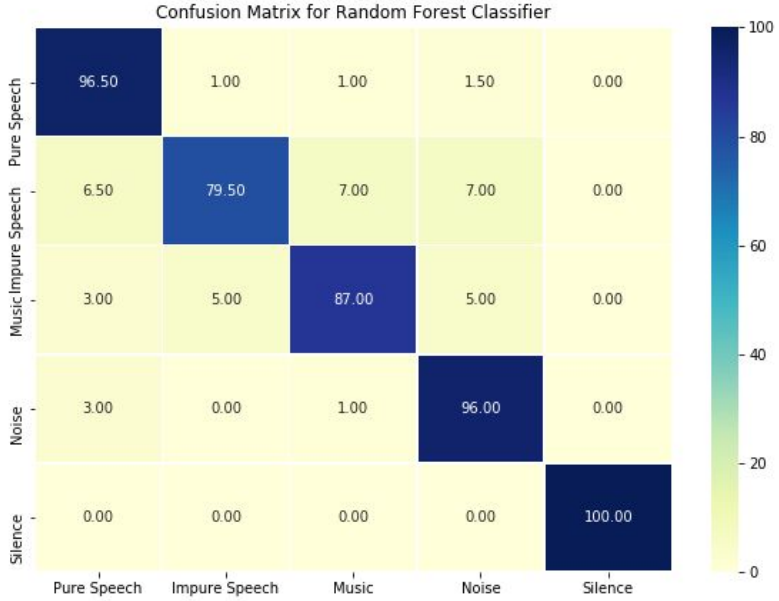

Figure 4. Random Forest Confusion Matrix

Our second form of results come from our real world audio tests. We selected a few clips from the internet that could act as basic tests. Our code plots support vector classifier predictions

and also rearranges the given audio clip so that it plays the audio in order of silence, noise, music, impure speech, pure speech. From our early results, it is apparent that much fine tuning must happen before the trained classifiers can handle data with transitions.

The first test handles classifying the first 30 seconds of the Taylor Swift song “Our Song.” This song starts off with a brief moment of interference (from the music video only), then there is music until the 12 second mark. At this point, Taylor Swift sings over a guitar.

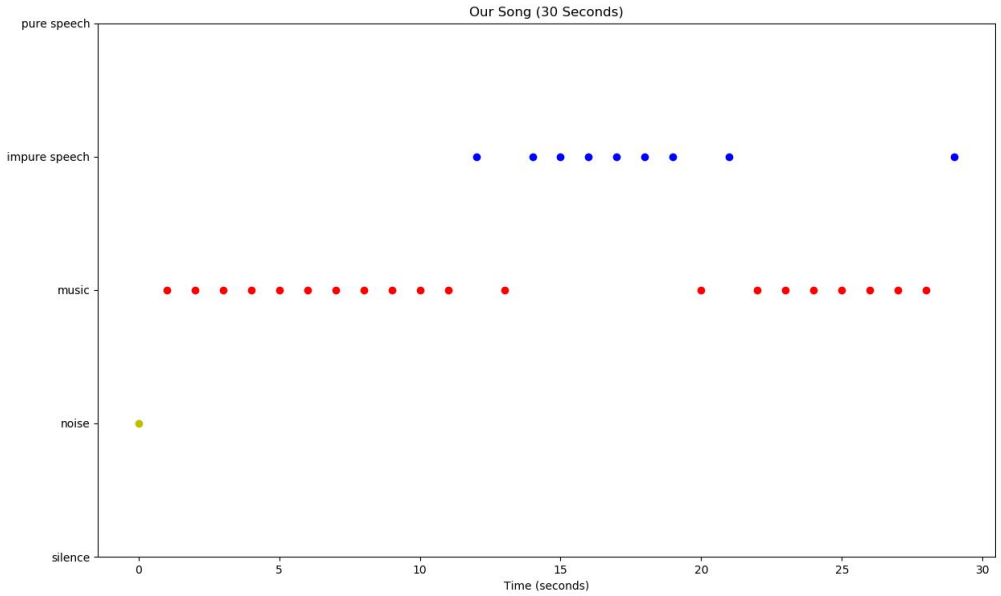

Figure 5: Results from classifying the intro of “Our Song”

The classifier correctly predicts the first 11 seconds of this clip and then notices the change to impure speech correctly. However, the musical classifications from ~20 seconds to the end are incorrect.

The second test shown handles the classifying of 30 seconds of the “God’s Plan” music video by Drake. This clip starts with a man talking and then at the 6 second mark, the music starts and Drake starts rapping intermittently over it. At the end, people are shouting wildly.

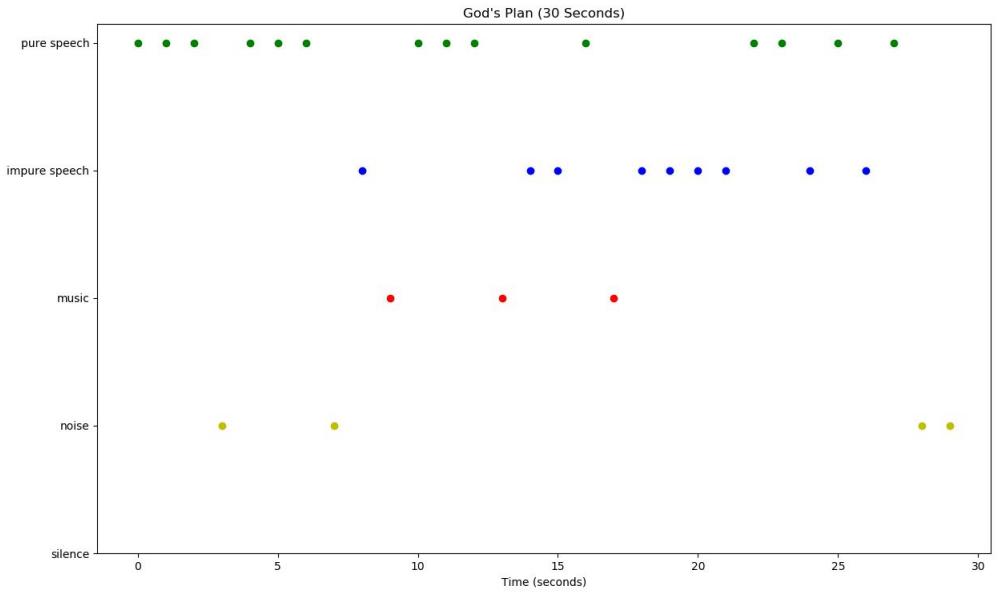

Figure 6. Results from classifying intro of song “God’s Plan”

This clips works well except for the pure speech classification after the 10 second mark, which are incorrect because the music is still playing.

4.1 Analysis

As can be seen from the data above, the support vector classifier produced the best results. The random forest classifier performed significantly better than the nearest neighbors classifier, but did not overtake the support vector. These results is expected. The support vector classifier performed very well in [6], achieving 95.8% accuracy, while the k-nearest neighbor algorithm only achieved 68.40% accuracy in the same report.

The random forest classifier, as expected as an ensemble classifier, achieved a high accuracy. The fact that the the support vector classifier outperformed the random forest classifier suggests that it is a very good candidate for classifying in this problem situation.

We were able to create a tool that automatically classifies real world audio data and can rearrange the audio automatically. However, the accuracy of the real world classifier was much lower than was to be expected after the results of the standard testing data. The transitions within the audio data caused trouble for the classifier. After analyzing the audio clips, it is apparent that this is because each frame (one second) of the real world data can contain multiple classes of audio. This means that the audio clips are significantly different from the ones that the classifier was trained on. Because of this difference, this result is disappointing but not surprising.

5 Conclusions and Future Work

The work we produced was able to nearly match the 95.8% accuracy achieved in [6]. Additionally, we reaffirmed that the k-nearest neighbors algorithm is a weak classifier for this problem space. The random forest classifier also performed as expected, and we showed that it is not as strong as the support vector classifier.

The real world testing did not perform as hoped, but we were able to successfully create a tool for classifying and rearranging arbitrary audio clips. The reasons behind the failures of the real world tester are apparent: classifying on audio that have transitions within the frames leads to low accuracy when the training data had no such transitions. This problem could be resolved in a number of ways. One approach could be to retrain the classifiers but only use 50 or 100ms frames, meaning there would be less frames with transitions within them. Another approach could be to use confidence of the support vector classifier. On one second frames that have a high confidence of the classification, the classifier would proceed as normal. When there is less confidence, this means that there is a high chance of a transition. On these frames, the frames could broken be into smaller frames in an attempt to identify the transition point.

This project provides additional evidence that the features extracted from the audio are useful for identifying the type of audio in the clip. Additionally, it supports that the support vector classifier is a strong candidate for handling this classification problem.

6 References

[1] Kemp, Thomas, et al. "Strategies for automatic segmentation of audio data." *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 00CH37100)*. Vol. 3. IEEE, 2000.

[2] Bietti, Alberto, Francis Bach, and Arshia Cont. "An online EM algorithm in hidden (semi-) Markov models for audio segmentation and clustering." *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015.

[3] Tzanetakis, George, and Perry Cook. "Multifeature audio segmentation for browsing and annotation." *Proceedings of the 1999 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics. WASPAA'99 (Cat. No. 99TH8452)*. IEEE, 1999.

[4] Zhang, Tong, and C-C. Jay Kuo. "Heuristic approach for generic audio data segmentation and annotation." *Proceedings of the seventh ACM international conference on Multimedia (Part 1)*. ACM, 1999.

[5] Patil, Vaishali V., and Pooja M. Gaud. "Improved speaker segmentation using clustering technique." *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*. IEEE, 2017.

[6] Lu, Lie, Stan Z. Li, and Hong-Jiang Zhang. "Content-based audio segmentation using support vector machines." *Proc. ICME*. Vol. 1. 2001.

[7] Gauvain, Jean-Luc, Lori F. Lamel, and Gilles Adda. "Partitioning and transcription of broadcast news data." *Fifth International Conference on Spoken Language Processing*. 1998.