SPEECH BASED PLAYLIST

A PROJECT REPORT

Submitted by

Ashwani Nadapana (174101006)

Priya Badchariya (174101047)

Puneet Raj Raipuria (174101045)

Baby Sai Sameera Salihundam (174101035)

Department of Computer Science Engineering

Indian Institute of Technology

Guwahati 781039 (India)

<u>ABSTRACT</u>

This project is based on Hidden Markov Model(HMM). It has been developed using C++ and Visual Studio (2010) IDE. The main goal of this project is to recognize the language spoken by the user and open the folder containing the playlist of the respective language.

Currently, the application supports the recognition of the languages namely Telugu, Hindi, English, Bengali and Tamil. It can be extended to support many other languages.

TECHNICAL DOCUMENTATION

PROPERTIES FILE:

"Recognition.properties" is the properties file of the project, where you mention all the properties used in the project.

The following are the properties used in the project .

1. *training_data_file_path:* This is the universe training vector file which is used to create codebook of size Kvalue

2. *record_file_path:* The path of the file which is created on recording the word spoken by the user using the recording module.

3. *cepstral_file_path:* The path of the cepstral coefficients file which are generated using Linear Predictive Coding(LPC) applied on recorded file.

4. *obs_sequence_file_path:* The path of observation sequence file which is created using cepstral coefficients file

5. *wav_file_path*: The path of the wave file which is created when the speech is recorded.

6. *batch_step*: The frame size used for calculating short term energies(STE) and zero crossing rate(zcr) of the speech signal.

7. *normalisation_amplitude*: The maximum value that could be present in the recorded file.

8. *record_input_duration*: The time duration of recording the speech.

9. *recording_module_exe_file_name*: The file path of recording module which is used to record the speech.

10. *Kvalue*: The prediction length used in Linear Predictive coding(P = 12).

11. *hamming_window_size*: The size of the frame which is used in performing Linear Predictive coding.

12. *frame_shift_size*: The no of samples that has to be shifted using sliding window transition in Linear Predictive coding(LPC)

13. *max_add_silence_frames*: The no of silence frames that has to be added for the trimmed recorded speech.

14. *K_levels*: The no of levels in the codebook.

15. *codebook_threshold*: The maximum distortion permissible in the codebook.

16. *splitting_factor_lbg*: The splitting parameter($0.01 <= \varepsilon <= 0.05$) used while splitting a level in codebook into two levels.

17. model_fileName: The name of the file which is used to save the re-estimated HMM model(λ) parameters.

18. *models*: The HMM models for each language which are used to recognize the word spoken. This parameter is in the format *<language>:<model_file_path>*

19. *model_spaces*: The no of spaces which are allowed in between α, β and γ parameters in the model file.

20. *songplaylistpath*: The path of folder which contains songs is placed against the respective language. This parameter is in the format *<language>:<songs_folder_path>* separated by ';'.

21. *model_creation_obs_seq_files*: The training data files which are used to train the HMM.

22. *codebookFilePath:* This is the path of the codebook.

OBJECTIVES OF USER DEFINED C++ AND .H FILES USED IN THE APPLICATION

SpeechBasedPlayList.cpp

This is the entry point of the application. Recognition of the spoken word is the main purpose of this .cpp file.

HMM.h

This header file is used for the following purposes.

1. Creating the model from utterances:

   Creates the re-estimated model from the training data files which is saved in model file, whose path is mentioned in the properties file. Training Data files are the sample files of different utterances of a word. In this project, we have trained the HMM using thirty utterances of each word.

2. Recognition of spoken word based on different models

   Scores each word model λ (A, B, PI) based upon the spoken word . Based on the scores, the word with the model which has the highest score is considered as the spoken word.

LBG.h

This is used to create the codebook of size KLevels from the universe training data set using **Linde–Buzo–Gray algorithm**(LBG) algorithm and is written into a file. This codebook is used to generate the observation sequence (indices of the best matched codebook level) for the recorded speech.

LPCProcessor.h

This helps to perform Linear Predictive Coding on the recorded file and generates the file which contains cepstral coefficients for the same. The generated cepstral coefficients are written to a file whose path is given in properties file.

ModelCreation.h

1. Creating the model from utterances:

   Re-estimates the model for a word, by continuously iterating the forward-backward procedure and Viterbi algorithm over the word's training dataset. The re-estimated model is saved in model file, whose path is mentioned in the properties file. *(Training dataset are the sample files of different utterances of a word. In this project, we have trained the HMM using thirty utterances of each word)*

RecordSplitter.h

This file is used to record the words(with some silence included in between) for the given input duration. Then, it splits the speech into different words, which in-turn are saved in different files.

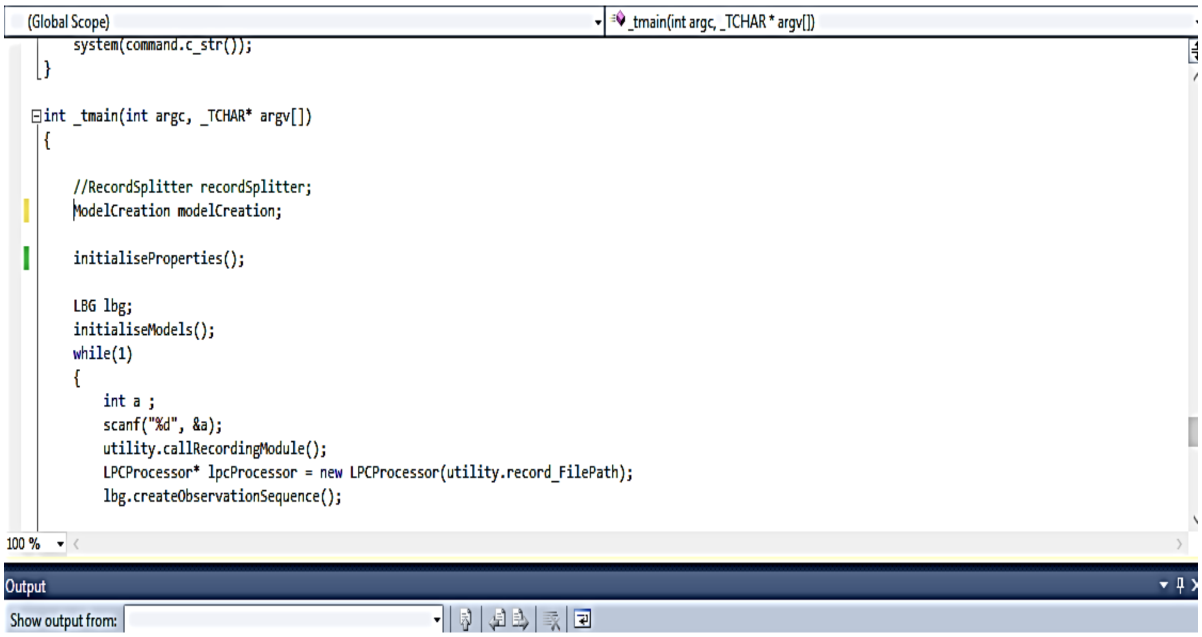
Utility.h

This utility helper class is used to perform some general functions like

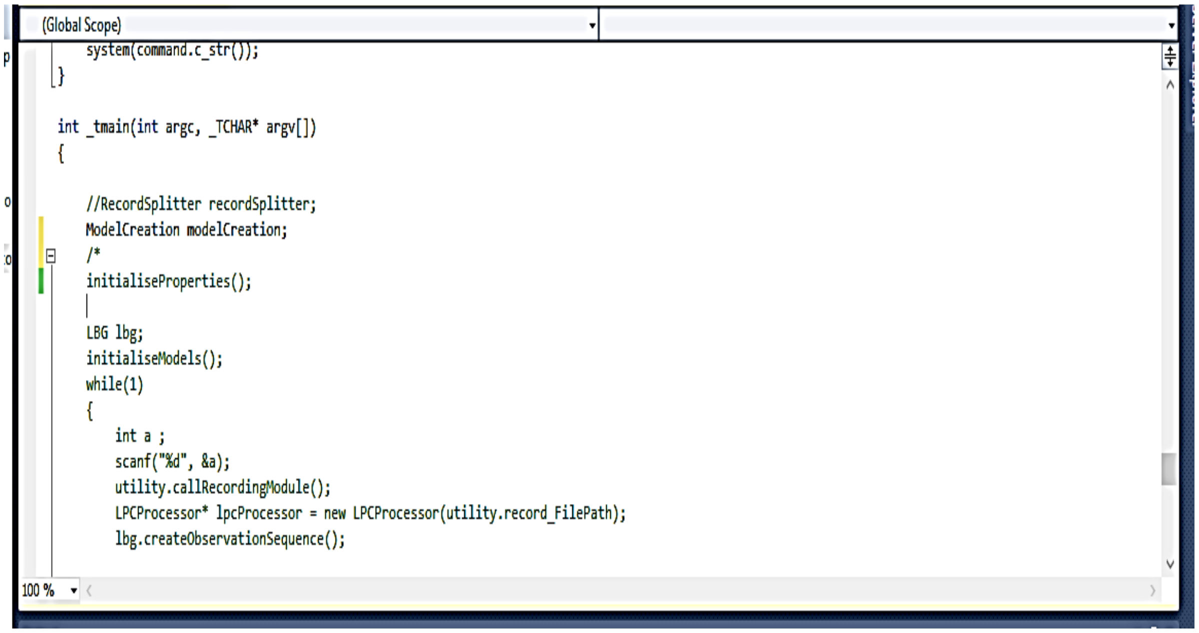
1. Recording the speech using recording module.
2. Calculate Tokhura distance between the cepstral coefficients of two vectors.
3. Checks the existence of a file.
4. Checks if a number is power of 2.

How to train the system with your own data (creating a model for a word)

1. Open SpeechBasedPlayList.cpp in Sources folder. Uncomment *" ModelCreation modelCreation*". Comment all other lines below it in main function. Save the file.

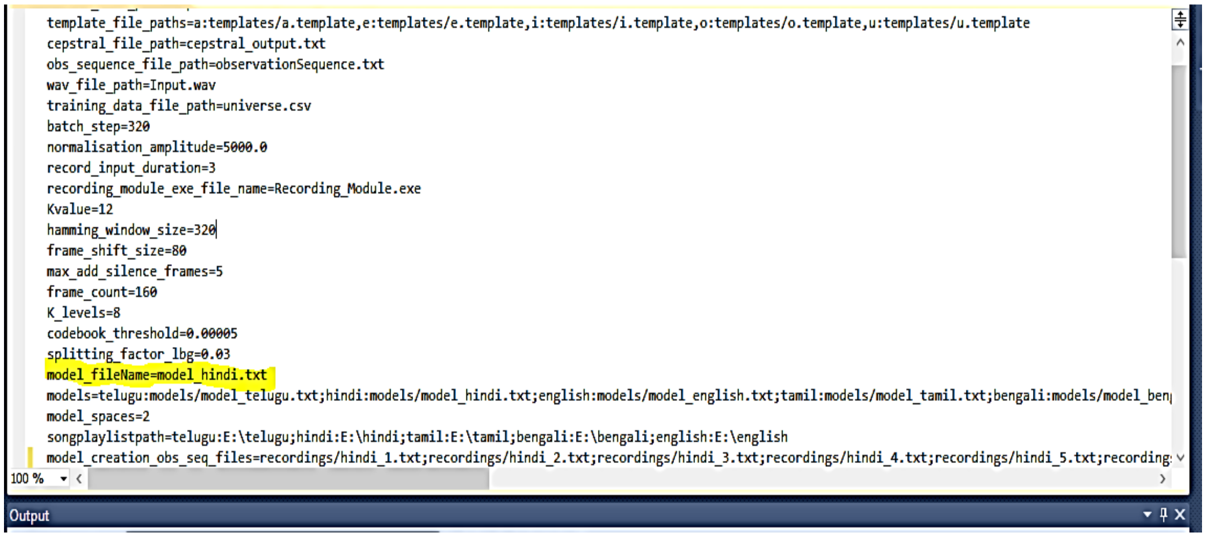


```
    system(command.c_str());
}

int _tmain(int argc, _TCHAR* argv[])
{

    //RecordSplitter recordSplitter;
    ModelCreation modelCreation;

    initialiseProperties();

    LBG lbg;
    initialiseModels();
    while(1)
    {
        int a ;
        scanf("%d", &a);
        utility.callRecordingModule();
        LPCProcessor* lpcProcessor = new LPCProcessor(utility.record_FilePath);
        lbg.createObservationSequence();
```



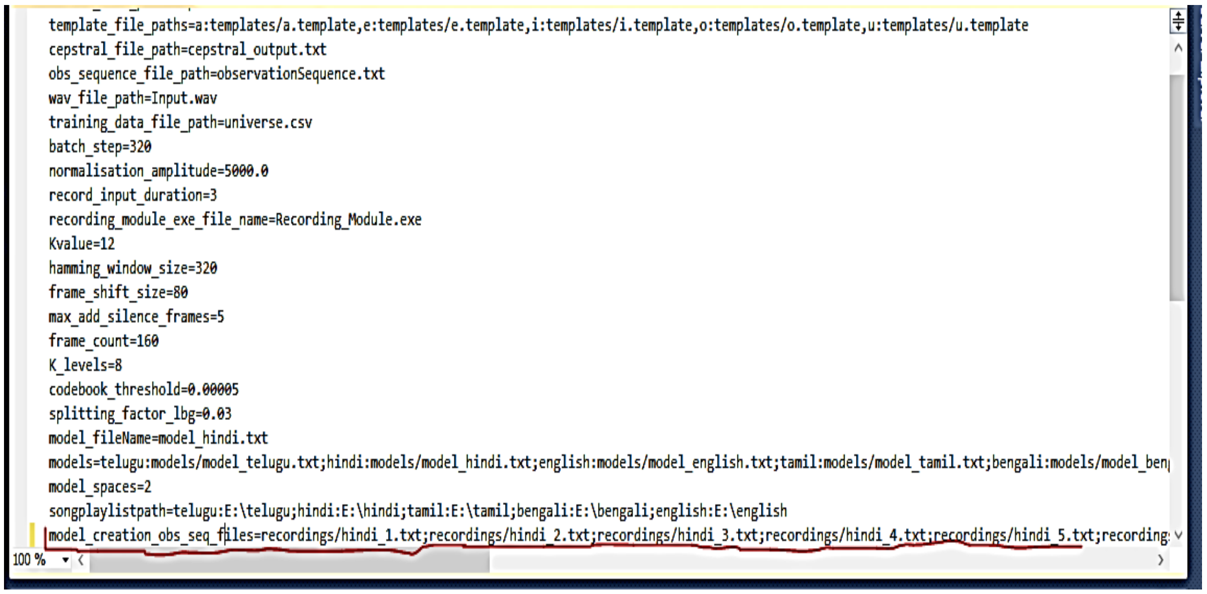
```
    system(command.c_str());
}

int _tmain(int argc, _TCHAR* argv[])
{

    //RecordSplitter recordSplitter;
    ModelCreation modelCreation;
    /*
    initialiseProperties();

    LBG lbg;
    initialiseModels();
    while(1)
    {
        int a ;
        scanf("%d", &a);
        utility.callRecordingModule();
        LPCProcessor* lpcProcessor = new LPCProcessor(utility.record_FilePath);
        lbg.createObservationSequence();
```

2. Open Recognition.properties file in Resource Files folder.
   2.1 Change value of model_filename to any desired name for model file.



```
template_file_paths=a:templates/a.template,e:templates/e.template,i:templates/i.template,o:templates/o.template,u:templates/u.template
cepstral_file_path=cepstral_output.txt
obs_sequence_file_path=observationSequence.txt
wav_file_path=Input.wav
training_data_file_path=universe.csv
batch_step=320
normalisation_amplitude=5000.0
record_input_duration=3
recording_module_exe_file_name=Recording_Module.exe
Kvalue=12
hamming_window_size=320
frame_shift_size=80
max_add_silence_frames=5
frame_count=160
K_levels=8
codebook_threshold=0.00005
splitting_factor_lbg=0.03
model_fileName=model_hindi.txt
models=telugu:models/model_telugu.txt;hindi:models/model_hindi.txt;english:models/model_english.txt;tamil:models/model_tamil.txt;bengali:models/model_ben
model_spaces=2
songplaylistpath=telugu:E:\telugu;hindi:E:\hindi;tamil:E:\tamil;bengali:E:\bengali;english:E:\english
model_creation_obs_seq_files=recordings/hindi_1.txt;recordings/hindi_2.txt;recordings/hindi_3.txt;recordings/hindi_4.txt;recordings/hindi_5.txt;recording
```

   2.2 Change the value of model_creation_obs_seq_files to the path names of all the training files for a word. Each two file paths must be separated by a semicolon.



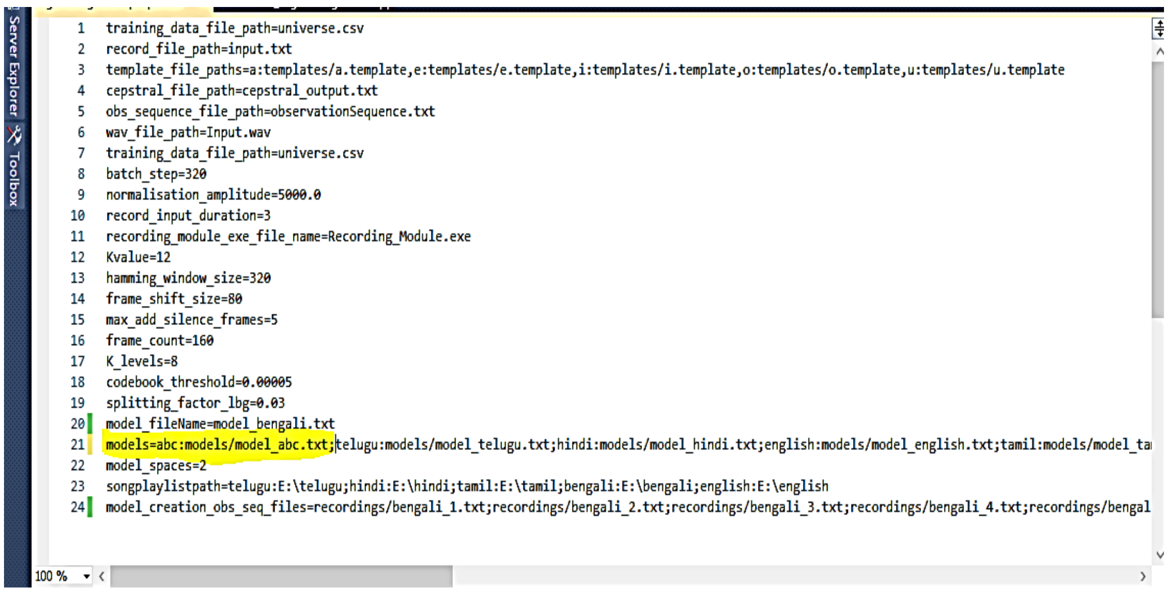
```
template_file_paths=a:templates/a.template,e:templates/e.template,i:templates/i.template,o:templates/o.template,u:templates/u.template
cepstral_file_path=cepstral_output.txt
obs_sequence_file_path=observationSequence.txt
wav_file_path=Input.wav
training_data_file_path=universe.csv
batch_step=320
normalisation_amplitude=5000.0
record_input_duration=3
recording_module_exe_file_name=Recording_Module.exe
Kvalue=12
hamming_window_size=320
frame_shift_size=80
max_add_silence_frames=5
frame_count=160
K_levels=8
codebook_threshold=0.00005
splitting_factor_lbg=0.03
model_fileName=model_hindi.txt
models=telugu:models/model_telugu.txt;hindi:models/model_hindi.txt;english:models/model_english.txt;tamil:models/model_tamil.txt;bengali:models/model_ben
model_spaces=2
songplaylistpath=telugu:E:\telugu;hindi:E:\hindi;tamil:E:\tamil;bengali:E:\bengali;english:E:\english
model_creation_obs_seq_files=recordings/hindi_1.txt;recordings/hindi_2.txt;recordings/hindi_3.txt;recordings/hindi_4.txt;recordings/hindi_5.txt;recording
```

3. Save Properties file.
4. Run the program.
5. A model file of name which you have provided earlier will be created.
6. Repeat steps 2-4 for creating models corresponding to each word.

How to test the system on live data

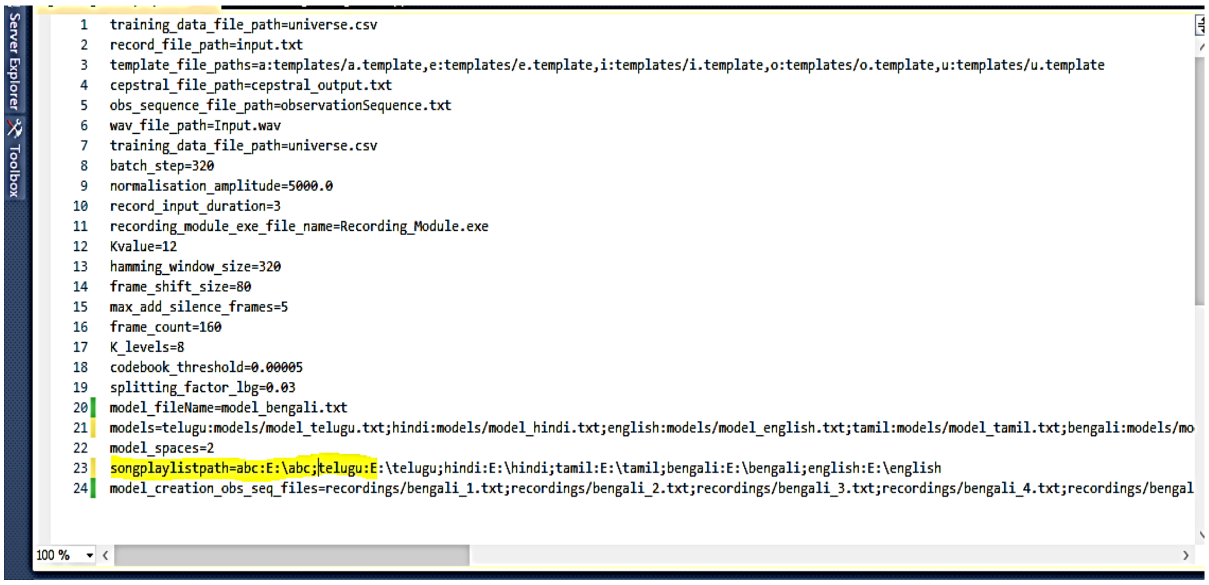
1. The models formed will be present in \SpeechBasedPlaylist\SpeechBasedPlaylist folder. Copy all those model files and paste them in \SpeechBasedPlaylist\SpeechBasedPlaylist \models folder.
2. Open Recognition.properties file in Resource Files folder.

   2.1 Change the value of attribute models as given in example: If the trained model for a word "abc" is in abc_model.txt. Then add following line as value of models, abc:models/abc_model.txt. Do this for each word. Separate values by semicolons.



```
training_data_file_path=universe.csv
record_file_path=input.txt
template_file_paths=a:templates/a.template,e:templates/e.template,i:templates/i.template,o:templates/o.template,u:templates/u.template
cepstral_file_path=cepstral_output.txt
obs_sequence_file_path=observationSequence.txt
wav_file_path=Input.wav
training_data_file_path=universe.csv
batch_step=320
normalisation_amplitude=5000.0
record_input_duration=3
recording_module_exe_file_name=Recording_Module.exe
Kvalue=12
hamming_window_size=320
frame_shift_size=80
max_add_silence_frames=5
frame_count=160
K_levels=8
codebook_threshold=0.00005
splitting_factor_lbg=0.03
model_fileName=model_bengali.txt
models=abc:models/model_abc.txt;telugu:models/model_telugu.txt;hindi:models/model_hindi.txt;english:models/model_english.txt;tamil:models/model_ta
model_spaces=2
songplaylistpath=telugu:E:\telugu;hindi:E:\hindi;tamil:E:\tamil;bengali:E:\bengali;english:E:\english
model_creation_obs_seq_files=recordings/bengali_1.txt;recordings/bengali_2.txt;recordings/bengali_3.txt;recordings/bengali_4.txt;recordings/bengal
```

   2.2 To add playlist corresponding to word edit the attribute songplaylistpath as sown in example: If the playlist path is E:\abc corresponding to the word abc then add following value to the attribute songplaylistpath as abc:E:\abc. Do this for each word. Separate values by semicolons.



```
training_data_file_path=universe.csv
record_file_path=input.txt
template_file_paths=a:templates/a.template,e:templates/e.template,i:templates/i.template,o:templates/o.template,u:templates/u.template
cepstral_file_path=cepstral_output.txt
obs_sequence_file_path=observationSequence.txt
wav_file_path=Input.wav
training_data_file_path=universe.csv
batch_step=320
normalisation_amplitude=5000.0
record_input_duration=3
recording_module_exe_file_name=Recording_Module.exe
Kvalue=12
hamming_window_size=320
frame_shift_size=80
max_add_silence_frames=5
frame_count=160
K_levels=8
codebook_threshold=0.00005
splitting_factor_lbg=0.03
model_fileName=model_bengali.txt
models=telugu:models/model_telugu.txt;hindi:models/model_hindi.txt;english:models/model_english.txt;tamil:models/model_tamil.txt;bengali:models/mo
model_spaces=2
songplaylistpath=abc:E:\abc;telugu:E:\telugu;hindi:E:\hindi;tamil:E:\tamil;bengali:E:\bengali;english:E:\english
model_creation_obs_seq_files=recordings/bengali_1.txt;recordings/bengali_2.txt;recordings/bengali_3.txt;recordings/bengali_4.txt;recordings/bengal
```

3. Save Recognition.properties file.
4. Run the module.
5. Enter any character to start the recording. Default input duration is 3 sec. Record in this duration and press Enter.
6. Our system will recognise the name of the language spoken and opens the folder containing playlist.