

Topics in Operating Systems (mini-project)

Open-set speaker recognition

by Ilya Kaganovsky

Abstract

Saya is a robotic receptionist of the Department of Computer Science in Ben-Gurion University of the Negev. Science 2006 students of the department make an effort to make Saya more human-friendly robot. They teach Saya to speak, to express emotions and more. My task is to teach Saya recognize a speaker by his voice. It may be very useful ability for receptionist.

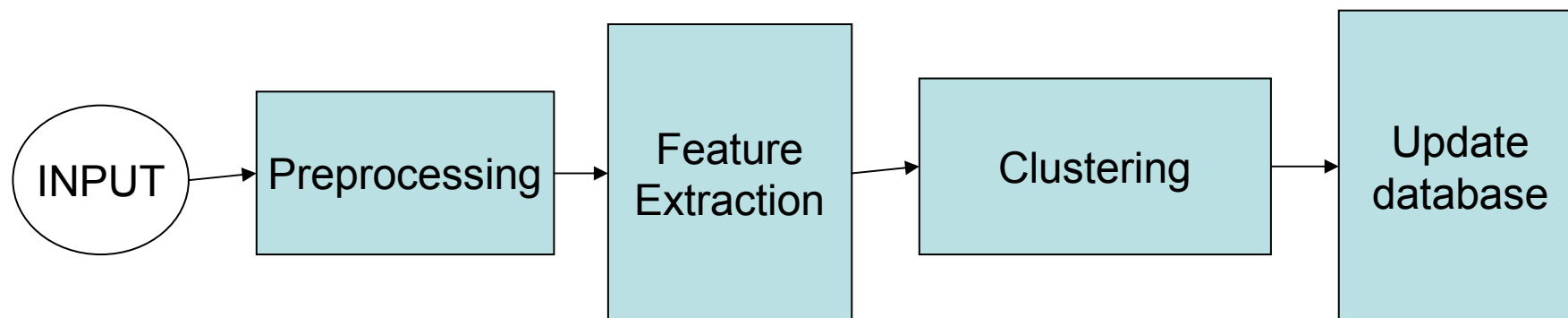
I use Modular Audio Recognition Framework, which is an open-source project initiated by four students of Concordia University, for implementing the open-set text independent speaker recognition application.

Definitions

- **MARF** is an open-source research platform and a collection of voice/sound/speech/text and natural language processing (NLP) algorithms written in Java and arranged into a modular and extensible framework facilitating addition of new algorithms.
- **Open-set recognition** is a recognition of a speaker, who is not necessarily has been shown before to the system (in the training part). Therefore the recognition system must be able to recognize a speaker as “unknown speaker”.
- **Text independent recognition** – the content of training and test utterances is any kind and it is unknown for the system.

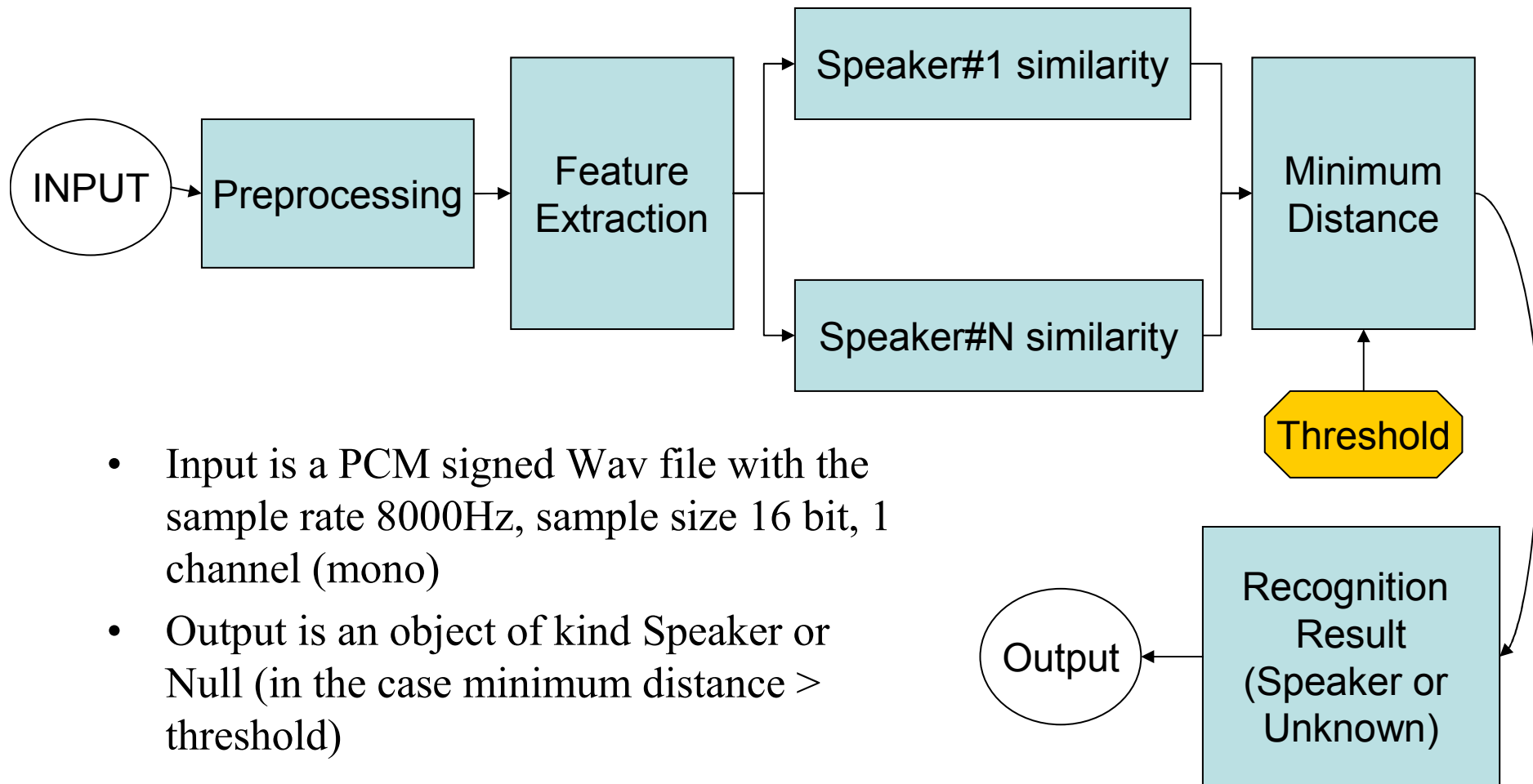
There are two main functions, train(..) and recognize(..). First is for training new speaker voice or upgrading voice data of existed in the database speaker. The second one is for recognizing speaker voice.

Train function (schematic)



- Input is a PCM signed Wav file with the sample rate 8000Hz, sample size 16 bit, 1 channel (mono)
- Note! If input file with the same name was trained already it will do nothing, even if the content of the file is different

Recognize function (schematic)



- Input is a PCM signed Wav file with the sample rate 8000Hz, sample size 16 bit, 1 channel (mono)
- Output is an object of kind Speaker or Null (in the case minimum distance > threshold)

Classes and functions

- **Speaker.java** – represent Speaker object. The speaker must have unique id and name.
- **NumGenerator.java** – simple class that help to generate unique numbers (is useful for generating different ids for all different speakers).
- **DB.java** – the purpose of the class is to manipulate with the database which will contain information about trained speakers.
 - *DB(String fileName)* is a constructor, fileName is a name of the database file. (simple text file where each raw formated as: <speaker id> <speaker name>
 - *remove(...)* delete the database file. And delete built by marf “marf.Storage.TrainingSet.*.gzbin” file which consist clustered feature extraction vectors of training sounds.

Classes and functions (cont.)

- *maxID()* returns the number of largest id in the database
- *getName(int pID)* find speaker with id pID in the database
- *connect()* – connecting to the database
- *query()* – loads data from the database
- *addSpeaker(Speaker oSpeaker)* – add speaker to the db
- *close()* – close the connection with the db
- **Main.java** – main class with the main functions
 - *train(), recognize(), setDefaultConfig()* – main functions
 - *validateVersions()* – make sure the application isn't run against older MARF version
 - *maxIdInDB()* – returns the largest speaker id in the database
 - *setDefaultConfig()* – sets default configuration parameters as endpoints for preprocessing, LPC for feature extraction and Chebishev distance for training and classification

Configuration parameters

- There are many different preprocessing, feature extraction and classification methods are implemented in MARF. I preferred to use:
 - Endpoint for preprocessing
 - Linear Predictive Coding (LPC) method for feature extraction
 - Chebyshev distance method for classification
- I had for that two reasons:
 - 1) This combination has shown good performance in the MARF developers test [2] (Recognition Rate 82.76%)
 - 2) Chebyshev distance shows here good difference in the cases of unknown and known speakers. If minimal distance $> 0.6-0.7$ we can be sure that the speaker is unknown for the system.

Suggestions for future work

The future task that should be done is adding functions `train()` and `recognize()` to the main Saya loop. It can be done in such way:

- If Saya recognizes some speech of kind “My name is Ilya”, it trains by voice of this speaker (the wav file must be saved previously).
- When somebody told Saya, she find his voice in database by recognize function, and can use his name in her speech. Moreover if she have message for this speaker, she will load it at once and will notify the speaker that she has for him message.

Another task that can be done is combination of speaker verification method with face verification, that can make the verification more efficient

References

- [1] <http://marf.sourceforge.net/> - marf website
- [2] <http://marf.sourceforge.net/docs/marf/0.3.0.6/report.pdf> – marf manual
- [3] <http://www.cs.bgu.ac.il/~orlovm/teaching/saya> – saya website
- [4] <http://www.speaker-recognition.org>
- [5] Magnus Nilson, Speaker Verification in JAVA, master thesis, School of Microelectronic Engineering Griffith University