University of Sheffield

Speaker Diarization System for the DIHARD Challenge

Mangesh Hambarde

Supervisor: Dr. Thomas Hain

A report submitted in fulfilment of the requirements
for the degree of MSc Computer Science with Speech and Language Processing

in the

Department of Computer Science

September 1, 2019

Declaration

All sentences or passages quoted in this report from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations that are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this project and the degree examination as a whole.

Name: ______________________________

Signature: ______________________________

Date: ______________________________

Abstract

Speaker Diarization is the task of finding out “who spoken when?” given an audio recording. It is an important field because it is a crucial preprocessing step for many areas in speech processing like speech recognition. Over recent years, the availability of fast computing power and emergence of massive amounts of multimedia data has boosted the field. The demand for various kinds of speech technology, and hence diarization, has become greater than ever. The performance of diarization systems has also improved significantly in the past decade or so thanks to the continued efforts of the research community. But even so, the absolute numbers tell a different story and it seems that there is still a long way to go. There is still a need for big improvements so that speaker diarization technology can be deployed in real world applications. This makes it an exciting research area which has a lot of potential to improve in the next few years.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Speaker diarization is the task of finding out "who spoke when?" in an audio recording with an unknown amount of speakers. It aims to find all segments of speech within the recording, possibly overlapping, along with their intra-recording speaker identities. It acts as an important upstream preprocessing step for most tasks in speech processing, like speech recognition, speech enhancement, speech coding etc.

With increase in computing power, speech processing technologies have achieved incredible advances in the past decade that were not possible earlier. This has increased interest in Rich Transcription (RT) technologies that can be used to automatically index the enormous amount of audio and video information that is generated in the modern world. Since speaker diarization is an important part in any RT system, there is a great deal of research interest in the area.

Diarization is not an easy problem since the output is affected by several factors like the application domain (broadcast news, meetings, telephone audio, internet audio, restaurant speech, clinical recordings etc), types and quality of microphones used (boom, lapel, far-field), inter-channel synchronization problems, overlapping speech, etc. These days, most of the research focuses on the meeting speech domain, since most problems that exist in speech recognition are encountered in this domain. The meeting scenario is thus often termed as "speech recognition complete".

The DIHARD challenge was created to establish standard datasets for diarization and create performance baselines for comparison, thus encouraging further research. The challenge focuses on "hard" diarization, combining several domains of speech like broadcast speech, meeting speech, telephone speech, and many more. Creating a system for the challenge can be a rewarding experience since it gives a chance to learn about state-of-the-art speaker diarization techniques.

1.1 Speaker Diarization

1.2 Motivation and Objectives

1.3 Report Outline

Chapter 2

Literature Review on Speaker Diarization

2.1 Introduction

2.2 Feature Extraction

2.3 Speech Activity Detection

2.4 Segmentation

2.5 Clustering

2.5.1 Speaker Representation

GMM

i-vectors

x-vectors

2.5.2 Agglomerative Clustering

2.5.3 Distance metrics

BIC

PLDA

2.6 Evaluation

Diarization Error Rate

Jaccard Error Rate

2.7 Kaldi toolkit

Chapter 3

DIHARD challenge setup

3.1 Task definition

3.2 Evaluation Tracks

3.3 Scoring

3.4 Datasets

Chapter 4

Baseline setup

4.1 Overview

There are three software baselines provided by the DIHARD II organizers, each for the parts of speech enhancement, speech activity detection and diarization. The speech enhancement baseline and the speech activity detection are meant to be used together in the case of system-generated SAD (tracks 2 and 4), but since we only work with reference SAD, we do not need them. Thus we will only describe the diarization baseline in the following sections.

The diarization baseline is based on the best performing submission [1] from John Hopkins University (JHU) in the previous year's DIHARD challenge (DIHARD I). There are 4 Kaldi recipes, each for an evaluation track, but we will focus only on the recipe for Track 1 since we only work with single channel audio and gold speech segmentation.

4.2 DIHARD datasets

4.3 Baseline directory structure

The baseline repository is localted at `https://github.com/iiscleap/DIHARD_2019_baseline_alltracks` and has the following directory structure. Some of the irrrelevant files have been removed.

```
DIHARD_2019_baseline_alltracks/
|-- data
|   |-- final.raw
|   |-- max_chunk_size
```

```
|   |-- min_chunk_size
|   |-- plda_track1
|   |-- plda_track2
|   |-- plda_track3
|   |-- plda_track4
|-- README.md
|-- recipes
|   |-- track1
|   |-- track2
|   |-- track2_den
|   |-- track3
|   |-- track4
|   `-- track4_den
|-- scripts
|   |-- alltracksrun.sh
|   |-- flac_to_wav.sh
|   |-- make_data_dir.py
|   |-- md_eval.pl
|   |-- prepare_feats.sh
|   |-- prep_eg_dir.sh
|   `-- split_rttm.py
`-- tools
    |-- env.sh
    |-- install_dscore.sh
    |-- install_kaldi.sh
}
```

The `data` directory has pre-trained models (in Kaldi binary format) and some configuration parameters - `final.raw` is the neural network x-vector extractor, and the `plda_*` files are the PLDA backends for the 4 tracks. The `recipes` directory has the `run.sh` files for all 4 recipes, we only care about `track1`. The scripts directory has extra scripts that are needed on top of the `egs/dihard_2018` Kaldi recipe - `alltracksrun.sh` is the main diarization script, `make_data_dir.py` makes the Kaldi data directory from the DIHARD datasets (creating files like wav.scp, segments, utt2spk etc), `prep_eg_dir.sh` copies the extra files from this repository to the `egs/dihard_2018` directory, `md_eval.pl` [2] is a diarization evaluation script that was developed by NIST, and others are self-explanatory. The `tools` directory holds scripts to install Kaldi and dscore [3], which are installed in the same directory.

The baseline code modifies and reuses the `egs/dihard_2018` recipe that was checked into Kaldi by the researchers at JHU. It does this by copying over new scripts and data that is needed to the `egs/dihard_2018` directory, `cd`'ing to that directory and running the recipe from there.

We modify and add scripts in this repository so we can easily run experiments with different parameters. The `run.sh` script is modified to allow easily changing parameters to run different experiments.

4.4 Initial segmentation

The initial segmentation step is done by `make_data_dir.py`. It deals with separating speech and non-speech segments from the recording files using the reference SAD which is provided in the form of HTK label files (.lab). Each audio recording has one label file. The label file has one line for each speech segment with the format <start-timestamp><end-timestamp>speech.

```
0.000 3.513 speech
4.698 7.133 speech
7.377 12.826 speech
13.284 16.797 speech
17.312 21.201 speech
...
```

This results in a bunch of segments which are known to be containing only speech. These are treated as "utterances" in Kaldi terminology and act as keys in the `utt2spk`, `feats.scp` and `segments` files. These files reside in two Kaldi "data directories", one for each dev and eval.

4.5 Features

The baseline then extracts 30 dimensional MFCC features for each of the every 10 ms using a 25 ms window. It uses the standard `steps/make_mfcc.sh` Kaldi script for this. The MFCC configuration used `mfcc.conf` is given below.

```
--sample-frequency=16000
--frame-length=25 # the default is 25
--low-freq=20 # the default.
--high-freq=7600 # Nyquist (8k in this case).
--num-mel-bins=30
--num-ceps=30
--snip-edges=false
```

Later, cepstral mean and variance normalization (CMVN) with a 3 second sliding window is applied using the `apply-cmvn-sliding` Kaldi tool.

4.6 Subsegmentation

After MFCC features are ready, the utterances are uniformly divided into smaller 1.5 second subsegments with a 0.75 second overlap. This creates new Kaldi data directories (one each for dev and eval sets) with newer keys corresponding to each subsegment. An x-vector is extracted from each of these subsegments in the next step using the Kaldi binary `nnet3-xvector-compute`.

4.7 Speaker representation

The baseline extracts an 512-dimensional x-vector from each subsegment using a neural network x-vector extractor. The extractor is trained on the datasets VoxCeleb I and II, along with added augmentation. Utterances smaller than 400 frames and speakers less than 8 utterances are discarded. Since the VoxCeleb dataset does not come with gold speech segmentation, the program `compute-vad` is used with the following configuration to classify each frame into speech or non-speech.

```
--vad-energy-threshold=5.5
--vad-energy-mean-scale=0.5
--vad-proportion-threshold=0.12
--vad-frames-context=2
```

It uses simple energy-based thresholding to generate a speech segmentation. Finally there are 1,277,503 utterances spoken by 7,351 speakers that can be used for training. Although the actual number is much more because of augmentation.

The augmentation is done by additive noise (noise, music, babble) using the MUSAN dataset and reverberation using the RIR dataset. The augmentation is done because it was determined in [4] that x-vectors exploit large quantities of training data much better than i-vectors, and show a significant increase in performance.

4.8 Scoring

For scoring two x-vectors, a PLDA backend is used as a distance metric. To train the PLDA backend, x-vectors are extracted from a random subset (size 128k) of the VoxCeleb dataset. To adapt the extracted x-vectors to the DIHARD domain, they are whitened with a whitening transform learned from the DIHARD development set. The PLDA model is trained using the x-vectors and the `ivector-compute-plda` Kaldi binary.

Each pair of x-vectors within a recording is then scored using the PLDA backend by reusing `score_plda.sh` from `egs/callhome_diarization`. These scores are stored as an affinity matrix for each recording.

4.9 Clustering

The x-vectors are then clustered using agglomerative hierarchical clustering (AHC) and a parameter sweep is done on the dev set to find the threshold that maximises the DER on the dev set. This threshold is then used for clustering the x-vectors of the eval set. The `agglomerative-cluster` Kaldi binary is used for clustering.

4.10 Diarization output

The clustering output is used to generate RTTMs using the script `make_rttm.py` from `egs/callhome_diarization`. The RTTMs give a flat segmentation of the recordings with no overlap. Since the x-vectors were extracted from segments that were overlapping, care needs to be taken when two adjacent segments are assigned to a different speaker. The script places the speaker boundary midway between the end of the first segment and the start of the second segment.

Chapter 5

Experiments and Results

5.1 Baseline results

5.2 Using Existing Pre-trained Models

5.2.1 Kaldi VoxCeleb x-vector model

5.2.2 Kaldi VoxCeleb i-vector model

5.3 Training Models on In-Domain Data

5.3.1 No Augmentation

5.3.2 Augmentation with Noise and Reverberation

5.4 Manual Speech Activity Detection

5.4.1 WebRTC

5.4.2 SHoUT toolkit

5.4.3 Energy-based

5.4.4 DNN-based

5.5 Speech Enhancement

5.5.1 Baseline

5.6 Alternative Distance Metrics for Clustering

5.6.1 BIC

5.6.2 Cosine

Chapter 6

Conclusions

Bibliography

[1] G. Sell, D. Snyder, A. McCree, D. Garcia-Romero, J. Villalba, M. Maciejewski, V. Manohar, N. Dehak, D. Povey, S. Watanabe *et al.*, "Diarization is hard: Some experiences and lessons learned for the jhu team in the inaugural dihard challenge," in *Proc. Interspeech*, 2018, pp. 2808–2812.

[2] "md-eval.pl." [Online]. Available: https://web.archive.org/web/20061001115045/http://www.nist.gov/speech/tests/rt/rt2006/spring/code/md-eval-v21.pl

[3] N. Ryant, "dscore." [Online]. Available: https://github.com/nryant/dscore

[4] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5329–5333.