Project 1: Product and Research Review

Identifying Multiple Speaker Using Speaker Diarization

Gilbert Yap

September 18, 2020
EC601 Product Design in Electrical and Computer Engineering
https://github.com/gilbertyap/EC601_Product_Design/tree/master/Project1

Introduction

There has been a rise in home consumer products that rely on human speech to operate since the release of Amazon's Amazon Echo in 2014. The Echo was Amazon's first product to feature their AI-powered voice assistant Alexa, which was designed to be more natural than voice assistants of the past by implementing natural language processing (NLP) and machine learning. Six years later, we now have voice assistant offerings and other vocal interaction features from other major companies including Amazon, Google, IBM, and Microsoft.

One particular area that researchers are focused on in AI and NLP is the identification and separation of multiple speakers in a single audio stream. The need for multiple speaker identification is apparent in traditional fields such as broadcast news, conference archiving, telephone logs, etc. [14]. An automatic and accurate transcription system could supplement audio and video recordings without the need for extra time dedicated to manual transcription. This is a major benefit to businesses that may need transcripts for support calls at call centers [8]. Early use of this technology can be seen today in automatic closed captioning of television programs. These early systems were aided by pre-written text [1] that may have partially provided context or information in the timing of captions. Today, researchers working on text-independent systems have benefitted from the advancements in machine and deep learning. These new learning models have launched a number of advancements in NLP and multiple speaker identification.

In this paper, I will first discuss the current field of speaker diarization, a framework for identifying and separating speakers within an audio stream. Afterwards, I will outline future applications and research that will continue to further the field. Then, I will detail the results of my own tests of open-source speaker diarization frameworks. The purpose of these tests is to use the systems and see the advantages and challenges of each in detail.

Speaker Diarization

In audio streams with multiple speakers, speaker diarization attempts to answer the question of "who spoke when". Speaker diarization is usually performed in four steps ([14], [1]): speech detection, speaker segmentation, feature extraction, and speaker clustering. Some systems have built-in evaluation methods, typically calculating Diarization Error Rate (DER). The steps of speaker diarization are not always performed in this order or as separate steps, but these four make up the core processes.

Speech Detection

The first challenge that a speaker diarization system is tasked with is broadly determining what sections of the audio stream are speech. An audio stream from broadcast news may contain music or commercials in addition to their speaking segments, and meetings may contain non-verbal sounds such as office noise. Speech Activity Detection (SAD) can be approached in a few ways: using thresholding, feature-extraction, or pre-trained learning models. Thresholding and feature-extraction methods hope to separate speech from non-speech using only signal characteristics. Thresholding is done by checking the energy level of the signal, and feature-extraction monitors spectral qualities to determine if the signal is speech. Pre-trained learning models check audio segments for similar features, but use these features to calculate the probability that they are speech based on a dataset.

Speaker Segmentation

The next step in a speaker diarization system is to break up the audio stream into time segments that only contain one speaker, which is sometimes called speaker change detection. There are a number of methods for how the speaker segments should be created, but the core principal behind them is the same. A window moves throughout the audio stream and determines if the audio in the current window can be modeled the same as what came before it or if it requires a new model. The commonly used methods for measuring this similarity is using the Bayesian Information Criterion [14] or cosine distance measurement [11]. These segments are not final and are subject to resegmentation after the initial clustering. Speaker segmentation can be challenging in some cases, such as situations where two speakers overlap.

Feature Extraction and Speaker Clustering

Feature extraction is focused on the retrieval of crucial speech characteristics that identifies a speaker from other sounds and speakers from one another. Features from a segment can be extracted in a number of ways such as calculating the Mel-frequency cepstral coefficients (MFCC) [10] or determining the eigenvoices [3] of a segment and comparing it to a pre-trained Gaussian Mixture Model (GMM). Prior to processing specific spectral information, some systems may take an optional first step of performing rudimentary feature extraction and clustering based on macro features [9] such as frequency bandwidth and spectral content pertaining to specific genders. This helps to initially filter distinct groups in an audio stream, which can then be clustered more accurately later. Speaker clustering will then take the features, or in the case of learning systems "embeddings", that were previously extracted and perform conventional clustering techniques on them. K-means, EM, and spectral clustering are some of the techniques used. Most speaker diarization systems will determine the number of clusters/speakers automatically, but under- or over-clustering may occur [1]. Some methodologies such as [3] specifically tested against two speakers only and concentrated on achieving high accuracy in segmentation and clustering. This would be more useful in standard telephone transcription, where there are typically only two speakers.

Optional: Evaluation

Speaker diarization performance is typically evaluated through the DER [14]. DER is calulated by taking the time measurement of false alarms, misses, overlaps, and mislabeling and dividing it by the entire length of the audio stream. These metrics can be calculated by referencing "ground-truth" information in supervised systems. "Miss" refers to the system having a speaker in the ground-truth but not in the processed audio. "False alarm" is the opposite of "miss". "Overlap" refers to segments of the audio stream that could not be labeled due to overlapping speakers. "Speaker error" refers to a mismatch between the generated segmentation/clsutering and the ground-truth reference. These four metrics can be weighed differently to give importance to different priorities such as correctly identifying the number of speakers or correctly identifying how long/often a specific speaker is detected.

Challenges in Speaker Diarization

While the question of "who spoke when" may seem simple, there are many challenges when trying to answer it. One such challenge is the quality of the audio recording. The recording of a conference or meeting typically uses a single far-field microphone or microphone array while a broadcast network will use a boom or lavalier microphone with a much higher signal-to-noise ratio. Other acoustic parameters that may affect the SNR are room reverberation and speaker volume. While the audio quality itself can confuse speaker diarization systems, so too can the content of the audio stream. For example, GMMs are not good at creating speaker segments based on short utterances and benefit from longer sections of speaking. Additionally, if speakers overlap or speak very quickly (e.g. natural conversation) then speaker models can become distorted and be placed into incorrect clusters [1]. Depending on the application, real time results may be required from a speaker diarization system, such as automatic captioning of live videos. Some systems were only designed for local processing and cannot be easily made into online systems.

Current Implementations

There are three major companies that provide speaker diarization as part of their speech processing systems. Amazon, Google, and IBM each have their own implementation that is available to developers through their cloud platforms. One unique thing about Google's implementation through Google Cloud Speech-To-Text is that it tags each word detected in the processed audio with a speaker ID number as opposed to segments. Google currently supports the most languages out of the three companies: English (Indian, British, American), Mandarin, French, German, Italian, Japanese, Portuguese (Brazil), Russian, and Spanish

(Spain) ([7], [5], [6]).

Of these three companies, Google is the only one to open-source part of their speaker diarization system. Their UIS-RNN [16] algorithm for segmenting and clustering is available on GitHub and was published as open-source in hopes that the community will be able to use their algorithm for improved speaker diarization systems. Google previously published an alternate version of the spectral clustering algorithm that they used in [15] to GitHub for developers to use in Python.

In addition to the many free and open-source speaker diarization systems and libraries, there are also groups providing multiple data sets that can be used to train speaker diarization models that rely on machine learning or neural networks. Two free datasets, The AMI Corpus and The ICSI Corpus, provide 70 and 100 hours respectively of recorded multi-speaker meeting speech along with annotations. The AMI Corpus is more robust than the ICSI Corpus and provides several microphone recordings from each meeting of different microphone types and setups.

Paths Forward

While implementations of machine learning have helped systems become more intelligent when making decisions regarding speaker identification in audio-only recordings, there has been other interdisciplinary research that shows how speaker diarization can be aided by visual and linguistic fields.

In [4], the Visual Geometry Group of University of Oxford proposed a diarization system that uses face recognition in tandem with speaker detection to more accurately label on-screen speakers in a variety of video program formats. This combination of audio and visual confirmation helps resolve some of the overlapped speaking issues and allows the system to separate overlapped speech into two distinct streams. This system uses a curated version of the "VoxConverse" dataset, a collection of "in the wild" videos, along with automatic and manually-verified transcriptions.

In [13], three Google engineers propose a combined automatic speech recognition (ASR) and speaker diarization system that focuses on linguistic and social verbal cues to create improved speaker labeling. The ASR and diarization systems are jointly processed by a recurrent neural network transducer that was trained on 15,000 hours of mainly two-speaker doctor-patient recordings of varying quality. The goal of the experiment was to correctly label the patient and the doctor within the recordings using only linguistic and audio cues.

Personally, I believe that linguistic clues could greatly help further the process of speaker diarization. As humans, we are able to detect and identify others based on a myriad of information from speech alone. Different people use different accents, place word stresses differently, or favor certain vocabulary. Of the speaker diarization systems outlined in this paper, most try to identify speakers strictly based on the voice's spectral contents, rather than the content or manner of their speech. It would also be interesting to see speaker diarization that takes into account speaker sentiment to help differentiate speakers who are portraying high levels of emotion in their speech.

Testing

Prior to writing this paper, I was interested in comparing two free, open-source Python libraries for speaker diarization. These two libraries were "pyannote-audio" and "pyBK". These libraries were chosen because of their differences in approaches to speaker diarization.

pyannote-audio

"pyannote-audio" [2] is a Python library for a supervised neural network that can be trained with a dataset to aid in speaker diarization. What makes "pyannote-audio" a powerful tool is that it can train its own speech activity detection, speaker change detection, overlapped speech detection, and re-segmentation systems. The system is also able to optimizate the diarization pipeline as a whole, which they claim other systems do not consider. They claim this leads to lower DER compared to other systems that have each system block trained individually and then connected without awareness of each other. Another unique feature of pyannote-audio is their use of SincNet [12] that allows their neural network to process waveforms directly instead of first generating MFCC or spectrograms for feature extraction. This lead to improvements across almost all tests when the system was trained with waveforms instead of MFCCs.

pyannote-audio provides pre-trained models through Torch Hub to get users started. These models have been trained on the AMI Corpus and the 2020 DIHARD Challenge datasets. While trying to use the provided pretrained models, I found that processing audio files larger than 30 seconds would take a much longer time viable for testing. Luckily, after checking the GitHub repository Issues section, I found that there were two other users with the same issue (the system was using $\geq$ 3GB of RAM to process audio files and caused stalling issues) and the issue was addressed by the developers. However, the pull request with the fix has not been approved as of September 17, 2020. As such, I was unable to proceed with testing pyannote-audio. It would have been good to compare pyannote-audio and pyBK directly to record the relative performance of each on the same audio files. pyannote-audio claims to be a more comprehensive speaker diarization system than pyBK, while pyBK claims to be lightweight and easily implemented into any system.

pyBK

"pyBK" [11] is a Python library that is able to perform speaker diarization without any need for external model training. The main component used within pyBK is the binary key background model (KBM). The entire audio file is used to train a newly generated Universal Background Model and set of "anchor models". This ultimately creates a set of gaussian mixture models that are used to produce a probability of likelihood that an audio frame matches a specific speaker. The KBM reduces audio frames into an N-dimensional binary vector that represents which GMMs the frame matches within the reference speaker models. This methodology is simple and lightweight, allowing it to be useful for online processing or low-resource environments.

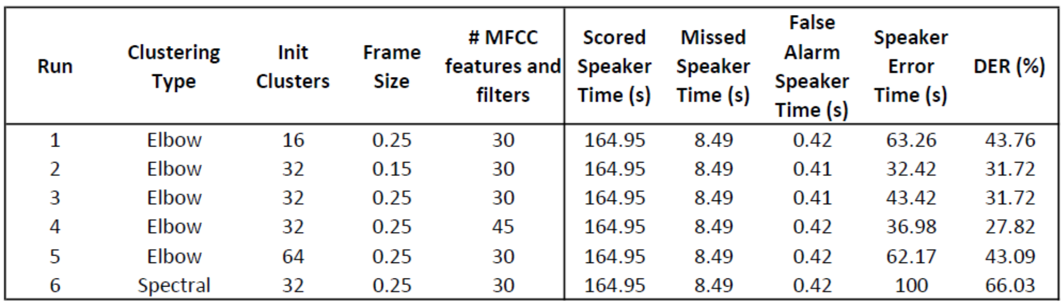
To test the performance of pyBK, a CNN panel discussion ("Student voter panel spars over gun control") was found on YouTube that contained multiple speakers that were human intelligible and had very little noise. The YouTube video was downloaded through a third party service as an .mp3 file with sample rate 44.1KHz and then converted into a .wav file with sampling rate 48KHz. The reason for converting into .wav with a higher sampling rate was due to the requirements of pyBK, which did not support the 44.1KHz sampling rate.

When the audio file is processed by pyBK, it outputs a Rich Transcription Time Marked (rttm) file that contains speaker ID along with the speaker start time and length. To determine how accurate the diarization process was, the output .rttm file needed to be compared against a "ground-truth" .rttm file. This file had to be created by hand since the YouTube video did not come with annotations, so the ground-truth file may had some amount of human error when determining the timestamps of each speaker. The audio sample was cut from its original 7 minutes 6 second length to 3 minutes to cut down on the amount of manual transcribing that was required by the evaluation system. The authors of the pyBK software library intend for users to fine-tune the initialization parameters present in a `config.ini` file to match their application. These optimizations include parameters such as the type of clustering to use, the number of MFCC features and filters to use, the number of initial clusters in the clustering process, and the size of the frame (in seconds) that is evaluated in the audio file. There are many other parameters that can be modified, but these were the ones tested as they were pereceived to have the highest impact on the DER.

Testing Results

Figure 1 belows shows the results of modifying the `config.ini` file that pyBK uses to initialize the system. Run 1 was the default configuration provided by the pyBK repository.

Run 4 had the best results compared to the default configuration, where the number of MFCC features and filters was increased from 30 to 45 and the number of initial clusters was doubled. Changing the frame size appeared to have no effect between Run 2 and 3 where all other parameters were held the same. However, the Speaker Error Time did increase by approximately 11 seconds. Run 6 had the worst performance, which used spectral clustering and double the initial clusters as Run 1. This resulted in a ∼37 second increase in Speaker Error Time and a DER worse than the original configuration in Run 1. The DER values here are much higher than the DER values (∼5% DER) the authors were able to produce on five other sample broadcast news files .



Run	Clustering Type	Init Clusters	Frame Size	# MFCC features and filters	Scored Speaker Time (s)	Missed Speaker Time (s)	False Alarm Speaker Time (s)	Speaker Error Time (s)	DER (%)
1	Elbow	16	0.25	30	164.95	8.49	0.42	63.26	43.76
2	Elbow	32	0.15	30	164.95	8.49	0.41	32.42	31.72
3	Elbow	32	0.25	30	164.95	8.49	0.41	43.42	31.72
4	Elbow	32	0.25	45	164.95	8.49	0.42	36.98	27.82
5	Elbow	64	0.25	30	164.95	8.49	0.42	62.17	43.09
6	Spectral	32	0.25	30	164.95	8.49	0.42	100	66.03

Figure 1. Table of six tests performed by modifying 4 parameters within pyBK. Speaker Error Time was the only metric that was more than marginally affected by modifying parameters within config.ini.

These results show that configuring the initialization parameters for pyBK mainly affected the assignment of speaker IDs to audio segments with this particular audio sample. Since Speaker Error Time was the only metric to be largely changed, it can be derived that detecting speech within the audio sample was not an issue, but rather assigning the identity of the speaker correctly was. Increasing the initial number of clusters from default to a certain extent gave more chances for the system to correctly differentiate the speakers. This effect was compounded when the number of MFCC were increased, since the number of features to compare within each segment was increased. Increasing the number of initial clusters further (as in Run 5 compared to Run 3) caused the DER to increase. Thus, a trend cannot be drawn that correlates the increases in feature extraction or clusters to improved DER.

Additionally, since only one audio sample was used, these conclusions cannot be made for all audio streams. The CNN panel discussion featured very clear audio since each speaker was given a lapel microphone inside of the CNN studio. There was no other background noise or non-speech sections in the audio that could confuse the system. Having such high DER values compared to the samples provided by pyBK seems like an issue of optimization or usage. In future tests, it would be beneficial to include a diverse set of audio clips with precise transcripts to further test the efficiency of the pyBK system.

Conclusion

As the world continues to move toward voice-interface systems, speaker diarization is becoming more and more important to both consumers and businesses. Thanks to modern day machine learning advancements, speaker diarization continues to improve and become part of everyday technologies. Automatic transcription and captioning services are able to run unsupervised thanks to vast data training sets, making broadcast news and online video captioning widely available. Google's effort to make their clustering and segmentation algorithms open-source has helped in the creation of new diarization systems for new platforms that can run online as opposed to on local systems. Project 1 has helped me gain a better understanding of how machine learning is being used in signal-related fields. Performing a deep-dive into speaker diarization research has showcased the importance of interdisciplinary research and how advancements in one field can propel another.

References

[1] X. Anguera et al. "Speaker Diarization: A Review of Recent Research". In: *IEEE Transactions on Audio, Speech, and Language Processing* 20.2 (2012), pp. 356–370.

[2] Hervé Bredin et al. "pyannote.audio: neural building blocks for speaker diarization". In: *ICASSP 2020, IEEE International Conference on Acoustics, Speech, and Signal Processing*. Barcelona, Spain, May 2020.

[3] Fabio Castaldo et al. "Stream-based speaker segmentation using speaker factors and eigenvoices". In: May 2008, pp. 4133–4136. DOI: `10.1109/ICASSP.2008.4518564`.

[4] Joon Son Chung et al. *Spot the conversation: speaker diarisation in the wild*. 2020. arXiv: `2007.01216 [cs.SD]`.

[5] *IBM Cloud Docs — Speech to Text — Output features*. Sept. 2020. URL: `https://cloud.ibm.com/docs/speech-to-text?topic=speech-to-text-output`.

[6] *Identifying Speakers*. Sept. 2020. URL: `https://docs.aws.amazon.com/transcribe/latest/dg/diarization.html`.

[7] *Language support — Cloud Speech-to-Text Documentation — Google Cloud*. Sept. 2020. URL: `https://cloud.google.com/speech-to-text/docs/languages`.

[8] Vidyasagar Machupalli. *Who's speaking? : Speaker Diarization with Watson Speech-to-Text API*. May 2017. URL: `https://www.ibm.com/cloud/blog/whos-speaking-speaker-diarization-watson-speech-text-api`.

[9] S. Meignier et al. "Benefits of prior acoustic segmentation for automatic speaker segmentation". In: *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 1. 2004, pp. I–397.

[10] H. Ning et al. "A spectral clustering approach to speaker diarization". In: *INTERSPEECH*. 2006.

[11] Jose Patino, Héctor Delgado, and Nicholas Evans. "The EURECOM submission to the first DIHARD Challenge". In: *Interspeech 2018, 19th Annual Conference of the International Speech Communication Association*. Hyderabad, India, Sept. 2018.

[12] Mirco Ravanelli and Yoshua Bengio. *Speaker Recognition from Raw Waveform with SincNet*. 2018. arXiv: `1808.00158 [eess.AS]`.

[13] Laurent El Shafey, Hagen Soltau, and Izhak Shafran. *Joint Speech Recognition and Speaker Diarization via Sequence Transduction*. 2019. arXiv: `1907.05337 [cs.CL]`.

[14] S. E. Tranter and D. A. Reynolds. "An overview of automatic speaker diarization systems". In: *IEEE Transactions on Audio, Speech, and Language Processing* 14.5 (2006), pp. 1557–1565.

[15] Quan Wang et al. *Speaker Diarization with LSTM*. 2017. arXiv: `1710.10468 [eess.AS]`.

[16] Aonan Zhang et al. *Fully Supervised Speaker Diarization*. 2018. arXiv: `1810.04719 [eess.AS]`.