Speaker Recognition

Evaluation of Top 2 Commercial Cloud APIs For Speaker Diarization

10.14.2020

Nicolas Stencel, Naga Sumanth, Andy Fausak, Daniel Mata

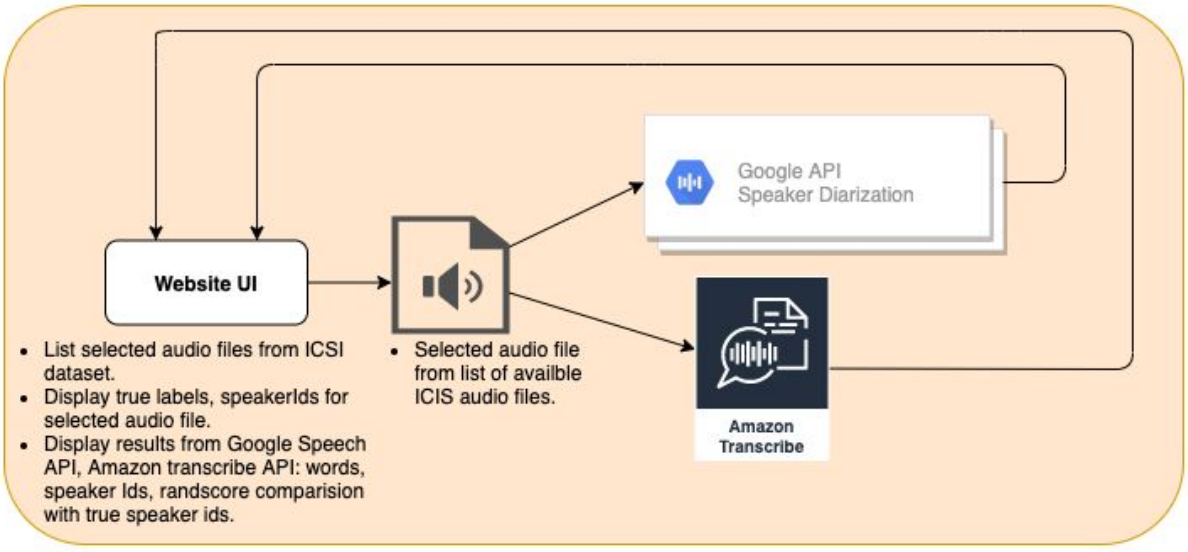
Project Abstract

Our project involves working with the speaker diarization process. Speaker diarization is a process in which audio is segmented into homogeneous segments that are based on speaker identity. One of its main use cases is optimizing automatic speech-to-text transcripts, and as the number of recordings begins to increase (ranging from lectures to broadcasts), research into speaker diarization has increased as well.

For the purpose of our project, we will be performing speaker diarization via two methods: using the Google Speaker Diarization API and then a trained model from this github. Once we have used both methods, we will compare the two results by using a Rand Index measure in order to observe the similarity.

After we write the API scripts that fetches the speaker diarization and transcription from both Google and Amazon we will incorporate it into the website. Once a user clicks the button for Google/Amazon API transcription on a particular audio. The website will dynamically take that audio, pass it to the API scripts and return the diarization and transcription of that audio to the user.

Overview



Project Name

Persona - Speaker Identification System

Team Members

Nicolas Stencel, Naga Sumanth, Andy Fausak, Daniel Mata

Roles

The project work was divided evenly amongst team members as such:

- Naga Sumanth - Website, Amazon API
- Daniel Mata - Google API, ICSI Dataset
- Nicolas Stencel - Website
- Andy Fausak - Debug/Tool Definition, Azure API

Weekly Meetings & Communication

We meet on the weekends, and on Tuesday to go over the items due the following day.

In order to communicate the on the fly, we've also created a discord server so if anyone ever needs help, another team member can hopefully provide some assistance.

Email

- Nicolas Stencel - nicolas.stencel@gmail.com
- Naga Sumanth - nagasumanth.ece@gmail.com
- Andy Fausak - andy.fausak@gmail.com
- Daniel Mata - mata.daniel777@gmail.com

Repository

Access to our Github can be found here. The Github contains all sample audio files we used.

Project Design and Milestones

Milestones

I. XML Parsing

ISCI Dataset: Our project relied on this dataset being our true label for comparison against the Google and Amazon API's. Our first step was to parse the many files contained with the dataset (each audio file was tied to several xml files for each individual speaker). This consisted of looping through each xml file for any given audio file, extracting the words and their corresponding speaker ID, and ordering based on overall timestamps.

Note: We only used nine 46-second audio files for the sake of time.

II. Speaker Diarization using APIs

Google API: This API needed a key downloaded as a json file, which would then be set as an environment variable named 'GOOGLE_APPLICATION_CREDENTIALS'. Once this was done, we were able to set the right parameters (*enable_speaker_diarization, diarization_speaker_count*) and obtain two arrays: a list of words and their corresponding speaker ID.

Amazon Transcribe API: Requires AWS CLI setup. AWS CLI needs to be configured with security keys that allows access to s3 bucket, amazon transcribe API. Audio files need to be uploaded to s3 bucket and that resource URL needs to be given as input to amazon transcribe API when requesting speaker diarization. Output is an URL that points to downloadable json. Json file needs to be parsed to extract required responses - such as words, time when words are spoken, speaker Ids.

Normalize Data / Audio (annotate) for processing

In order to make our comparisons work, we needed to normalize our dataset. Our Google API, for example, would return a list of speaker ID's with the 'first' speaker being labeled as '3' (e.g. [3,3,3,1,1,1,1,1,1,0,0,0]) as opposed to our true labels (e.g. [0,0,00,1,1,1,2,2,,2,2,0,0,3]). Thus, we relabeled our API output's based on initial speaker appearance.

III. Rand Score Index Comparison Utility

Using the *sklearn* library, we were able to import *adjusted rand score index* and compare the true labels against each normalized API's output by simply passing the two lists of speaker IDs as arguments. From here, the rand score would be calculated and a score ranging from -1 to 1 would be returned based on similarity.

IV. Integration

Before setting up a virtual environment, we pip installed *sklearn* and *google-cloud*. Then, using *Flask*, we set up our environment and were able to successfully integrate our 9 audio samples, the corresponding true labels, the corresponding API outputs, and the rand score for each API output.

Resources and Related Projects:

Inspiration: Automatic Speaker Recognition

- Tutorial detailing how training data was used for speech recognition

Implementation: Google Cloud-Speech-to-Text API

- This API will serve as the first method in performing our speaker diarization

Implementation: Amazon Transcribe API

- This API will serve as the second method in performing our speaker diarization

Data: ICSI Corpus

- This link provides access to roughly 70 hours of recorded lectures, our true labels will be extracted from here

Consultation: Tabshum, Tasina **<ThasinaTabashum@my.unt.edu>**

- Tasina has experience obtaining annotated audio as well as a diarization background. She also provided links to audio data for examples.

PERSONA - Evaluation of Top 2 Commercial Cloud APIs For Speaker Diarization

—

Nicolas Stencel, Naga Sumanth, Andy Fausak, Daniel Mata

Overview

- According to Wiki, “Speaker Diarization” is defined as, “The process of partitioning an input audio stream into homogenous segments according to the speaker identity...”
- *Persona* compares a “True Label” diarization to cloud diarization services, and displays a “Rand Score” to indicate the accuracy of the cloud diarization to the *True Label Reference*.
- *True Label Reference*, consists of an ICSI WAV audio file, text of words spoken (including speaker id).

Method

- 9 diarization *True Label Reference* samples from International Computer Science Institute (ICSI) are compared to alternate diarization API's (Google/Amazon):
  - For a given Diarization API, a WAV file is selected for translation to diarization representing words spoken and speaker ID.
  - 2 Lists are created: one comprises words spoken; the other contains corresponding speaker ID.
- Lists are compared to *True Label Reference* and a *Rand Score* is computed.
- Standard Deviation and Mean are maintained as tests are repeated to indicate consistency and similarity of diarization methods.

[1] How are you guys doing?

[2] Can't complain

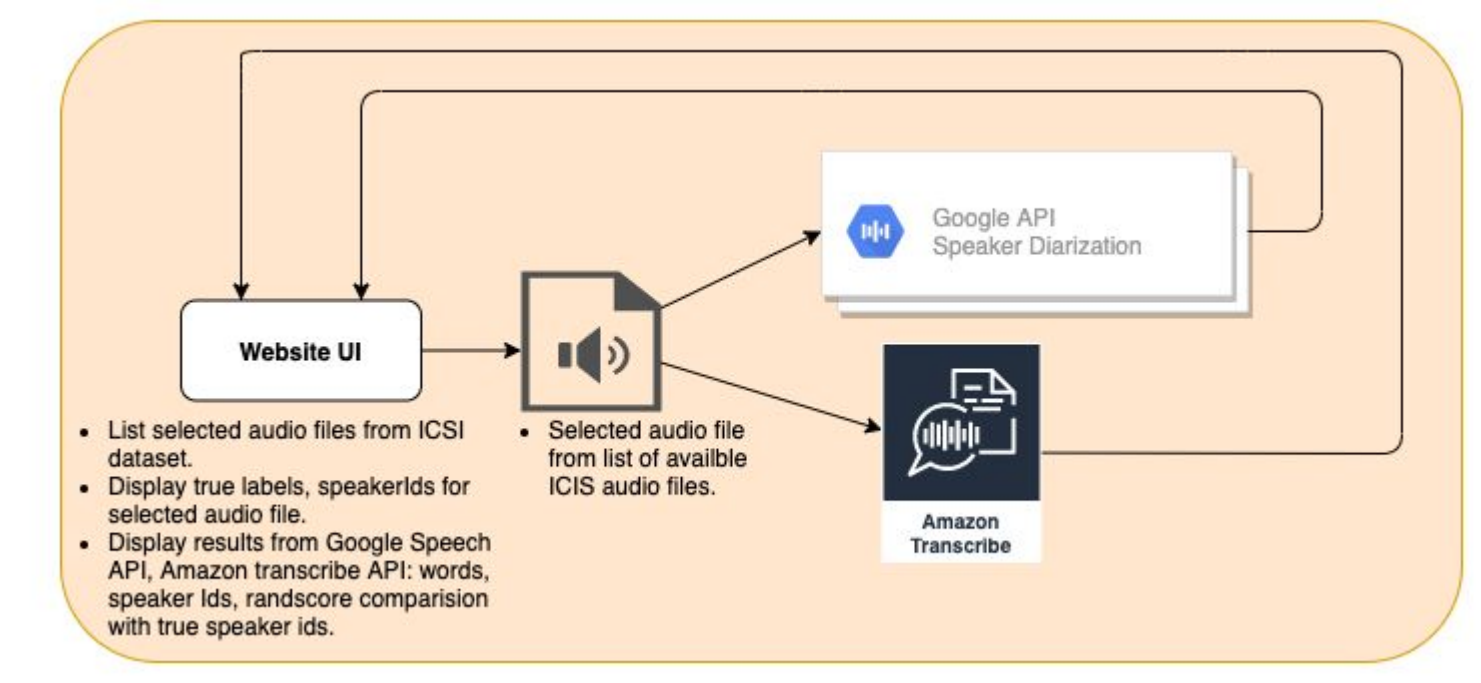
[3] Same. What about yourself?

[1] I'm fine thank you.

Technologies Used

- Website - Flask
- Collaboration - Github
- API Scripts - Python
  - **Python packages and libraries (remember to ‘pip install’ these libraries in your virtual environment! The website will not run and will throw an error otherwise!)**
    - Sklearn (pip install sklearn)
    - Google (pip install google)
    - Google-cloud (pip install google-cloud)
    - aws cli (pip install aws cli)
      - **Once you install these packages, deactivate and then reactivate your virtual environment before running**
  - Google and Amazon API keys. Right now they are masked on Github. If you want to extend this project, let us know we can guide you how to setup keys for your cloud accounts.

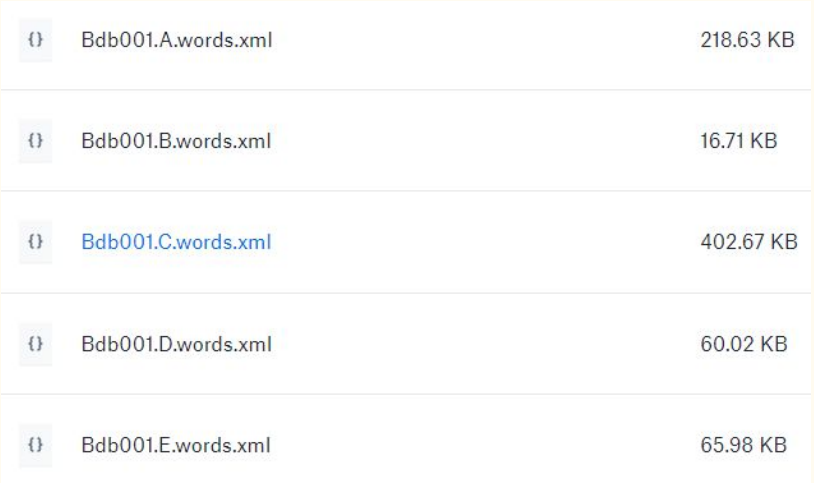
Project Overview Continued



Website Demo:

True Label Extraction - XML Parsing

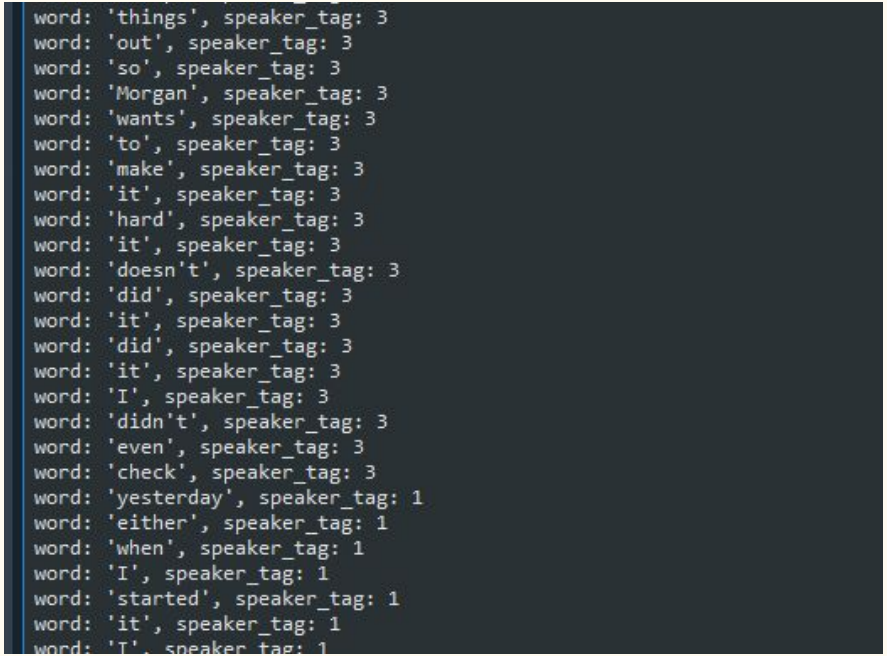
- For each WAV file:
  - Individual XML files for each speaker
  - All XML files were contained in ‘Words’ folder
- Words filtered using c=”W” attribute
- Words were extracted using ‘starttime’ and ‘endtime’ attributes
  - Both were in seconds
  - Timestamp range based on audio sample




```
11 <w nite:id="Bdb001.w.691" starttime="164.014" endtime="164.334" c="W">Oh</w>
12 <w nite:id="Bdb001.w.692" starttime="164.334" endtime="164.334" c="CM">,</w>
13 <w nite:id="Bdb001.w.693" starttime="164.334" endtime="164.454" c="W">I</w>
14 <w nite:id="Bdb001.w.694" starttime="164.454" endtime="164.834" c="W">remember</w>
15 <w nite:id="Bdb001.w.695" starttime="164.834" endtime="165.204" c="W">seeing</w>
16 <w nite:id="Bdb001.w.696" starttime="165.204" endtime="165.264" c="W">an</w>
17 <w nite:id="Bdb001.w.697" starttime="165.264" endtime="165.644" c="W">example</w>
18 <w nite:id="Bdb001.w.698" starttime="165.644" endtime="165.734" c="W">of</w>
19 <w nite:id="Bdb001.w.699" starttime="165.734" endtime="165.974" c="W">this</w>
20 <w nite:id="Bdb001.w.700" starttime="165.974" endtime="165.974" c=".">.</w>
21 <w nite:id="Bdb001.w.701" starttime="165.974" endtime="166.184" c="W">Yeah</w>
22 <w nite:id="Bdb001.w.702" starttime="166.184" endtime="166.184" c=".">.</w>
```

Google Speech to Text API

- ICSI Dataset
- Needed to enable Speaker Diarization
- Parameters we worked on:
  - Sample_rate_hertz
  - Enable_speaker_diarization
  - Diarization_speaker_count
- Default output from API



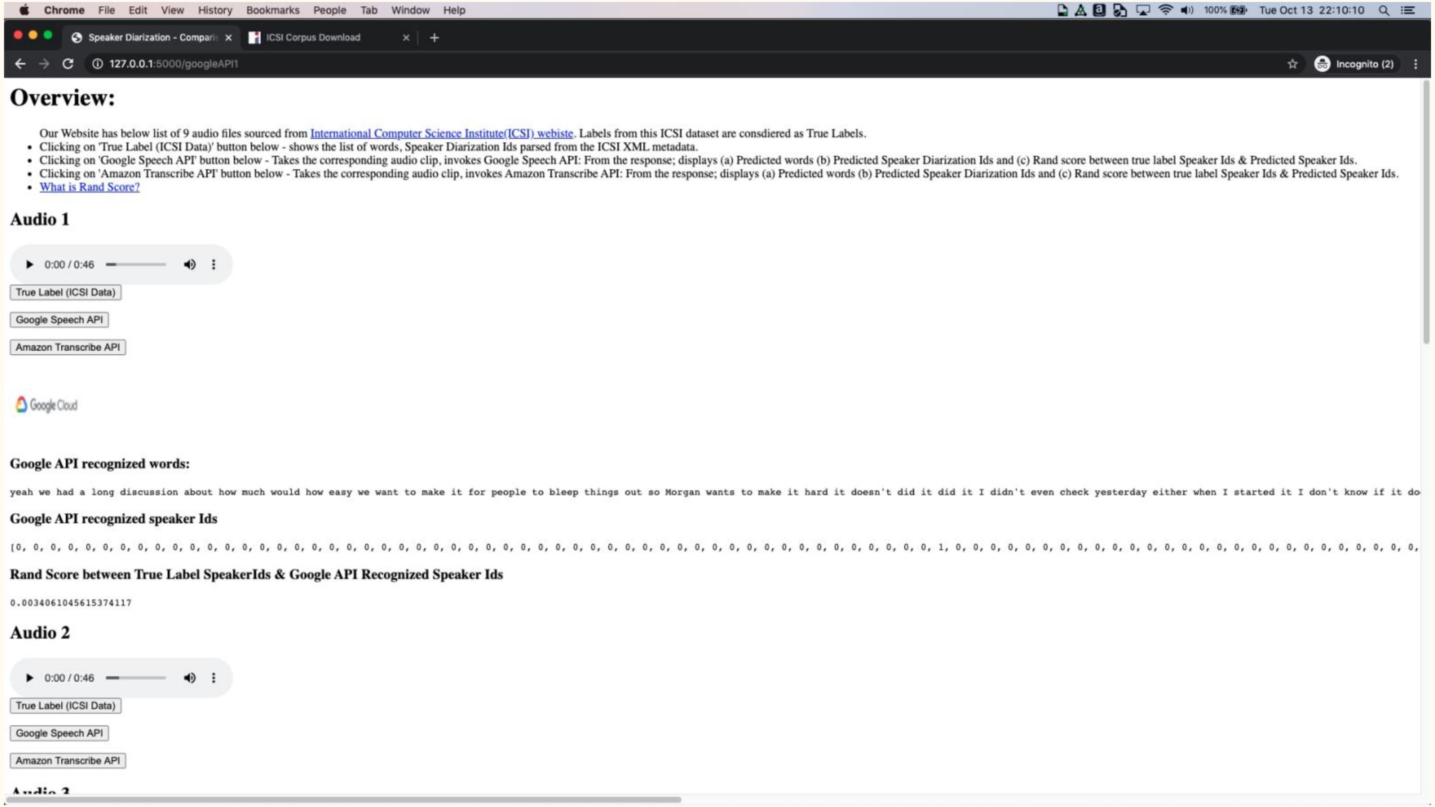
Google Speech to Text API, Normalization

- Needed to normalize our labels for Rand Score comparison
- Label based on order of initial speaker appearance

```
[3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 3, 3, 3, 3, 3, 3, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 3, 3, 3]

Normalized Labels: {3: 0, 1: 1}
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0]
```

Google API Display on the Website

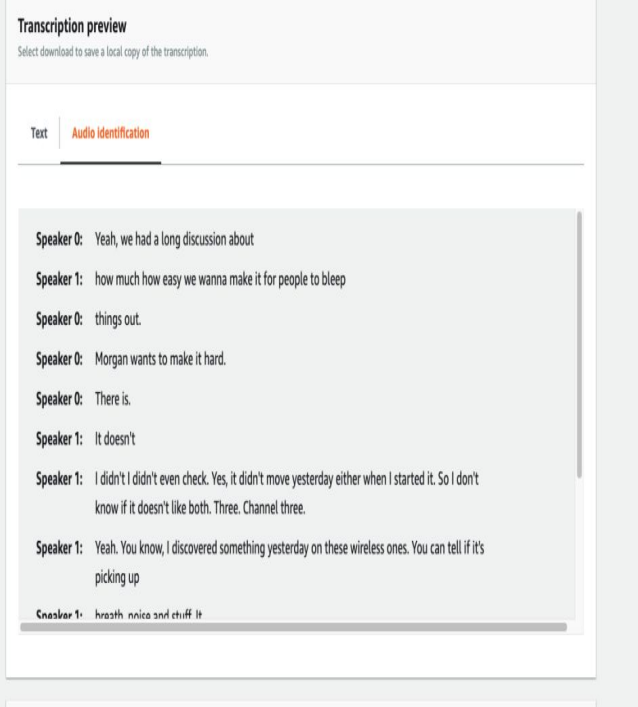


Outputs: True Label transcript, AWS Console

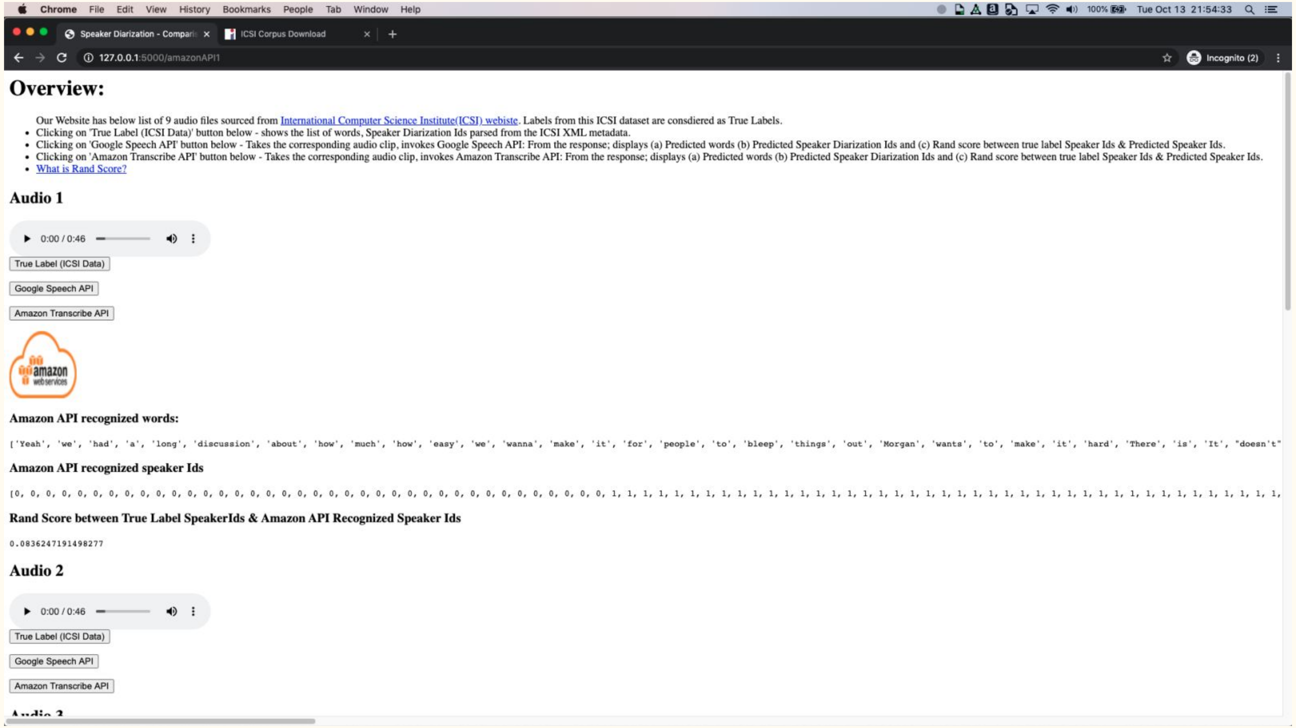
```
</Preamble>


<Transcript StartTime="0.0" EndTime="3021.968">

  <Segment StartTime="0.000" EndTime="3.956" Participant="me018">
    <NonVocalSound Description="mike noise"/>
  </Segment>
  <Segment StartTime="0.056" EndTime="1.861" Participant="me011">
    Yeah, we had a long discussion about
  </Segment>
  <Segment StartTime="2.674" EndTime="6.315" Participant="me011">
    how much w- how easy we want to make it for people to bleep things out. So -
  </Segment>
  <Segment StartTime="5.100" EndTime="7.881" Participant="me018">
    <NonVocalSound Description="mike noise"/>
  </Segment>
  <Segment StartTime="6.260" EndTime="6.864" Participant="fe016" CloseMic="false">
    Right.
  </Segment>
  <Segment StartTime="6.315" EndTime="7.061" Participant="me011">
    <VocalSound Description="breath"/> <NonVocalSound Description="mike noise"/>
  </Segment>
  <Segment StartTime="7.516" EndTime="8.757" Participant="fe016" CloseMic="false">
    O_K. So this is -
  </Segment>
  <Segment StartTime="8.169" EndTime="11.100" Participant="me011">
    Morgan wants to make it hard. <VocalSound Description="laugh"/>
  </Segment>
  <Segment StartTime="8.320" EndTime="13.210" Participant="me018">
    <NonVocalSound Description="mike noise"/>
  </Segment>
  <Segment StartTime="9.751" EndTime="13.549" Participant="fe016" CloseMic="false">
    The, uh, counter is not <Pause/> moving again. It -
  </Segment>
```



Amazon API Display on the Website



Rand Score

- Imported from sklearn
- Two arguments passed in:
  - True labels Speaker Ids
  - API labels Speaker Ids
- An adjusted rand score was returned

```
>>> from sklearn.metrics.cluster import adjusted_rand_score
>>> adjusted_rand_score([0, 0, 1, 1], [0, 0, 1, 1])
1.0
>>> adjusted_rand_score([0, 0, 1, 1], [1, 1, 0, 0])
1.0
```

Labelings that assign all classes members to the same clusters are complete be not always pure, hence penalized:

```
>>> adjusted_rand_score([0, 0, 1, 2], [0, 0, 1, 1])
0.57...
```

ARI is symmetric, so labelings that have pure clusters with members coming from the same classes but unnecessary splits are penalized:

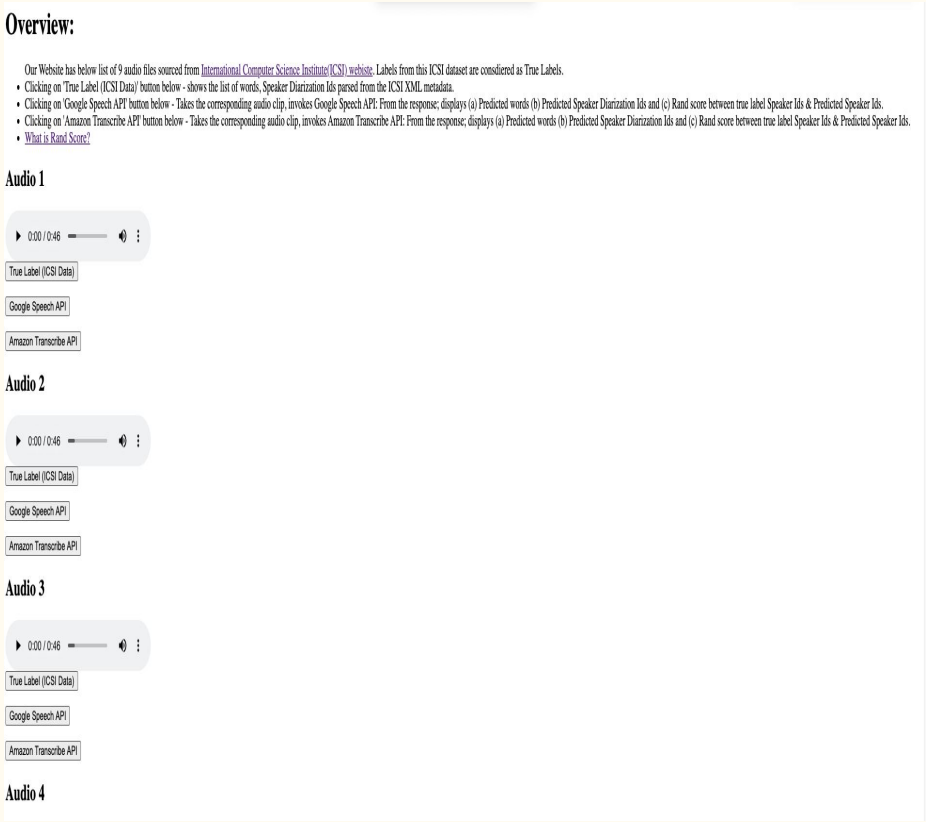
```
>>> adjusted_rand_score([0, 0, 1, 1], [0, 0, 1, 2])
0.57...
```

If classes members are completely split across different clusters, the assignment is totally incomplete, hence the ARI is very low:

```
>>> adjusted_rand_score([0, 0, 0, 0], [0, 1, 2, 3])
0.0
```

What has been done

- Parsed True labels information from ICSI dataset transcript XML file.
- Obtained the Amazon Speaker diarization from Amazon API, and dynamically displayed it on our site.
- Computed Rand Score calculation between:
  1. True Label, Google API
  2. True Label, Amazon API
- Website has 9 .wav files displayed on the website. Users can listen to the audio files, get their true value transcription, and also get the speaker diarization and speech transcription of the 2 APIs (Google and Amazon)



What can be expanded

- Add Azure API functionality. Right now we only have the Google and Amazon APIs working.
- Have the users be able to upload their own audio files to be transcribed rather than relying on files that have already been uploaded with a “true” value transcription and diarization.
- Right now our project is pretty static, only transcribing 9 audio files. Extend it into a full blown tool that can be used to transcribe and compare different audio files
- Get better accuracy on the XML parsing. The current XML parsing is based on conditions that we wrote that sometimes leave out some word.
- Better website UI that is aesthetically pleasing.