IMPROVING AUTOMATIC TRANSCRIPTION USING NATURAL LANGUAGE PROCESSING

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Anna Kiefer

March 2024
COMMITTEE MEMBERSHIP

TITLE: Improving Automatic Transcription using Natural Language Processing

AUTHOR: Anna Kiefer

DATE SUBMITTED: March 2024

COMMITTEE CHAIR: Foaad Khosmood, Ph.D.
Professor of Computer Science

COMMITTEE MEMBER: Alexander Dekhtyar, Ph.D.
Professor of Computer Science

COMMITTEE MEMBER: Franz Kurfess, Ph.D.
Professor of Computer Science
ABSTRACT

Improving Automatic Transcription using Natural Language Processing

Anna Kiefer

Digital Democracy is a CalMatters and California Polytechnic State University initiative to promote transparency in state government by increasing access to the California legislature. While Digital Democracy is made up of many resources, one foundational step of the project is obtaining accurate, timely transcripts of California Senate and Assembly hearings. The information extracted from these transcripts provides crucial data for subsequent steps in the pipeline. In the context of Digital Democracy, upleveling is when humans verify, correct, and annotate the transcript results after the legislative hearings have been automatically transcribed. The upleveling process is done with the assistance of a software application called the Transcription Tool. The human upleveling process is the most costly and time-consuming step of the Digital Democracy pipeline. In this thesis, we hypothesize that we can make significant reductions to the time needed for upleveling by using Natural Language Processing (NLP) systems and techniques. The main contribution of this thesis is engineering a new automatic transcription pipeline. Specifically, this thesis integrates a new automatic speech recognition service, a new speaker diarization model, additional text post-processing changes, and a new process for speaker identification. To evaluate the system’s improvements, we measure the accuracy and speed of the newly integrated features and record editor upleveling time both before and after the additions.
ACKNOWLEDGMENTS

Thanks to:

- My advisor, Foaad Khosmood, and committee members, Alex Dekhtyar, and Franz Kurfess
- The entire Transcription Tool team past and present, including Toshihiro Kuboi, Hans Poschman, Srirag Vuppala, Mihika Nigam, Thorsten Ruprechter, and Christine Robertson
- The Digital Democracy team at Cal Poly, especially Thomas Gerrity and Kenny Lau
- CalMatters and the Knight Foundation, Arnold Ventures, and the Lodestar Foundation for their funding and support
- The maintainers of the many software packages and tools this thesis employs, especially the open source ones
- My parents for encouraging my educational pursuits
- My husband Aakash Ahamed for his continued love and support of my many endeavors
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>LIST OF TABLES</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF FIGURES</td>
<td>xi</td>
</tr>
</tbody>
</table>

## CHAPTER

1 Introduction ................................. 1 
1.1 Digital Democracy ....................... 2 
1.1.1 Project History ....................... 3 
1.1.2 Project Architecture ................. 4 
1.2 Methodology ............................... 6 
1.2.1 Results ................................ 8 
1.2.2 Organization ......................... 9 
2 Background ................................. 10 
2.1 Transcription Tool ...................... 10 
2.2 Software ................................ 13 
2.3 Natural Language Processing .......... 14 
2.4 Machine Learning ......................... 15 
2.5 Evaluation Metrics ...................... 16 
3 Related Work ................................. 21 
3.1 Other Transcription Editing Tools ...... 21 
3.2 Transcription Tool ....................... 22 
3.3 Natural Language Processing .......... 23 
3.3.1 Automatic Speech Recognition ....... 24 
3.3.2 Named Entity Recognition ............. 26
6.3 Speaker Identification ............................................. 106
6.4 Other Changes ......................................................... 106
6.5 Total Cost Improvement ............................................ 107
7 Future Work ............................................................. 108
  7.1 Improved Video Transcription Pipeline ....................... 108
    7.1.1 Faster Speaker Diarization .................................... 108
    7.1.2 Additional ASR Services ....................................... 109
    7.1.3 More Inclusive Speaker Identification ....................... 109
    7.1.4 Improved Fault Tolerance ..................................... 109
    7.1.5 Multilingual STT ................................................ 110
  7.2 Automatic Data Refreshes ......................................... 110
  7.3 Faster Speaker Alignment ......................................... 111
BIBLIOGRAPHY ............................................................. 112
APPENDICES
  A Evaluation of each AST Service per Transcript .................. 125
  B Actual vs Predicted Speaker Counts of Pyannote across Videos . . .  127
<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Main Thesis Contributions</td>
<td>7</td>
</tr>
<tr>
<td>4.1 Transcription Tool Versions</td>
<td>33</td>
</tr>
<tr>
<td>4.2 STT Comparison Video Information</td>
<td>37</td>
</tr>
<tr>
<td>4.3 Sample of the NER Data</td>
<td>50</td>
</tr>
<tr>
<td>5.1 Person and Organization Entities Found</td>
<td>71</td>
</tr>
<tr>
<td>5.2 Runtime Comparison</td>
<td>72</td>
</tr>
<tr>
<td>5.3 Transcription Cost</td>
<td>74</td>
</tr>
<tr>
<td>5.4 Metadata for Comparison Video</td>
<td>81</td>
</tr>
<tr>
<td>5.5 Diarization Results for Test Video</td>
<td>81</td>
</tr>
<tr>
<td>5.6 Diarization Runtime for Test Video</td>
<td>82</td>
</tr>
<tr>
<td>5.7 Runtime of PyAnnote for Different AWS Instances</td>
<td>84</td>
</tr>
<tr>
<td>5.8 Possible Classification Outcomes for NER Tagging</td>
<td>85</td>
</tr>
<tr>
<td>5.9 Evaluation Metrics for NER Models</td>
<td>88</td>
</tr>
<tr>
<td>5.10 Possible Classification Outcomes for Speaker Identification</td>
<td>90</td>
</tr>
<tr>
<td>5.11 Metrics for Token-Based Matcher, Full Dataset (47K Utterances)</td>
<td>91</td>
</tr>
<tr>
<td>5.12 Metrics for QA Model with Names Included (Craig et al., 2023)</td>
<td>92</td>
</tr>
<tr>
<td>5.13 Metrics for QA Model with Names Excluded (Craig et al., 2023)</td>
<td>93</td>
</tr>
<tr>
<td>5.14 Metrics for QA Model, Full Dataset (47K Utterances) (Craig et al., 2023)</td>
<td>94</td>
</tr>
<tr>
<td>5.15 Comparison Metrics of Tool Versions</td>
<td>97</td>
</tr>
<tr>
<td>5.16 New Transcriber Metrics across Versions</td>
<td>101</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Diagram of Digital Democracy Resources (Khosmood, 2023)</td>
<td>5</td>
</tr>
<tr>
<td>2.1</td>
<td>Transcription Tool Edit Transcript View</td>
<td>11</td>
</tr>
<tr>
<td>2.2</td>
<td>Transcription Tool View Status of Hearing Administrator Page</td>
<td>12</td>
</tr>
<tr>
<td>4.1</td>
<td>Transcription Tool Data Pipeline (Kuboi, 2023)</td>
<td>30</td>
</tr>
<tr>
<td>4.2</td>
<td>Activity Diagram of the Transcription Tool Pipeline</td>
<td>32</td>
</tr>
<tr>
<td>4.3</td>
<td>Video Metadata from the 10 Longest Videos</td>
<td>39</td>
</tr>
<tr>
<td>4.4</td>
<td>Diagram for Transcription Request Process</td>
<td>40</td>
</tr>
<tr>
<td>4.5</td>
<td>Diagram of the Speaker Diarization Process</td>
<td>45</td>
</tr>
<tr>
<td>4.6</td>
<td>Diagram of the Speaker Identification Process</td>
<td>48</td>
</tr>
<tr>
<td>4.7</td>
<td>Example of NER Tagging using SpaCy</td>
<td>49</td>
</tr>
<tr>
<td>4.8</td>
<td>Three Token Patterns used for Speaker Identification</td>
<td>53</td>
</tr>
<tr>
<td>4.9</td>
<td>BERT Model Architecture (Devin, 2018)</td>
<td>54</td>
</tr>
<tr>
<td>4.10</td>
<td>A Sample of the Nickname Dataset used to Train the QA model (Craig et al., 2023)</td>
<td>56</td>
</tr>
<tr>
<td>4.11</td>
<td>Validation Loss Curves for Multiple Experiments Demonstrating Necessity of Answer in Context (Craig et al., 2023)</td>
<td>58</td>
</tr>
<tr>
<td>4.12</td>
<td>Validation Loss Curve for the Final Model with Names Included (Craig et al., 2023)</td>
<td>58</td>
</tr>
<tr>
<td>4.13</td>
<td>Validation Loss Curve for the Final Model with Names Omitted (Craig et al., 2023)</td>
<td>59</td>
</tr>
<tr>
<td>4.14</td>
<td>An Example Utterance Row Containing a Speaker Introduction</td>
<td>60</td>
</tr>
<tr>
<td>4.15</td>
<td>Speaker Identification Modal Window</td>
<td>60</td>
</tr>
</tbody>
</table>
4.16 Create New Speaker Modal ........................................... 61
4.17 Utterance UI before the Cut Button is Selected .............. 63
4.18 Utterance UI after the Cut Button is Selected .............. 64
4.19 Utterance UI after the End Time Edit ......................... 64
4.20 Diagram of the Process to Update Person IDs for Subsequent Tasks 66

5.1 WER, MER, and WIL for each Service ......................... 68
5.2 The Cosine Similarities Computed for each Service .......... 69
5.3 Histogram of transcription runtime as a % of video time. .... 73
5.4 A Selection of Five Utterances from the Transcripts .......... 75
5.5 The Transcript Request Parameters as of March 2024 ......... 77
5.6 The Transcription Tool’s AssemblyAI Usage Over Time ...... 79
5.7 The Transcription Tool’s AssemblyAI Costs Over Time ...... 80
5.8 Predicted vs. Actual Speaker Count for Videos ............... 83
5.9 Confusion Matrix for the Person Entity Tagging for SpaCy .. 85
5.10 Confusion Matrix for the Person Entity Tagging for Flair ... 86
5.11 Confusion Matrix for the Person Entity Tagging for BERT .. 86
5.12 Confusion Matrix for the Person Entity Tagging for NLTK .. 87
5.13 Confusion Matrix for the Person Entity Tagging for Stanford NLP. 87
5.14 Average Time to Run NER for a Single Utterance .......... 89
5.15 Confusion Matrix for the Token Matcher ....................... 91
5.16 Confusion Matrices Produced for the Final Model when Testing with Names Included and Names Excluded (Craig et al., 2023) .......... 92
5.17 Confusion Matrix when the QA model is Run on the Full Dataset of 47K utterances. ................................................. 94
5.18 A Sample of the Speakers Identified by the QA Model. ...... 96

xii
5.19 Average Transcription Time in Minutes for Version 0 . . . . . . . . 99
5.20 Average Transcription Time in Minutes for Version 1 . . . . . . . . 100
5.21 Average Transcription Time in Minutes for Versions 2 and 2.5 . . . 101
5.22 Average Editing Time for New Transcribers for Version 0 . . . . . . 102
5.23 Average Editing Time for New Transcribers for Version 1 . . . . . . 103
5.24 Average Editing Time for New transcribers for Versions 2-2.5 . . . 103

A.1 WER, MER, and WIL for AssemblyAI and Amazon Transcribe . . . 125
A.2 WER, MER, and WIL for Deepgram and Whisper . . . . . . . . . 126

B.1 Percent Difference between Pyannote’s Predicted Speakers Compared to Ground Truth Actual Speaker Count . . . . . . . . . . . . . . 127
In the United States, power is shared between federal and state governments. With the exception of Nebraska, state legislatures are divided into upper and lower houses. In the state of California, the upper house is called the Senate and the lower house is called the State Assembly. These two governing bodies are responsible for enacting laws and providing a wide range of services at the state level, including law enforcement, education, health and human services, the allocation of funding and resources, infrastructure maintenance, and many others. Managing these services is often done through legislation proposed as bills that are discussed in committee hearings.

Held by the legislature, these hearings represent the main way to debate and craft legislation in a parliamentary system. They also offer valuable information for citizens to understand government decisions. In the United States, government records are transcribed at the federal level and made publicly accessible, but this is not the case with legislative hearings at the state level. Democratizing and disseminating government information at the state level may lead to a more informed citizenry and help hold government officials accountable. A 2015 poll by the Institute for Advanced Technology & Public Policy (IATTP) at California Polytechnic State University San Luis Obispo (Cal Poly) shows overwhelming public support for increased transparency of the state legislature, including requiring video recordings of legislative hearings online within 24 hours and all public documents to be online and searchable (Myers, 2015). This holds true regardless of political belief or party representation.
Unfortunately, the information output from the state government is often disorganized, unstructured, and ill-suited for public accessibility (Khosmood et al., 2014). While there are some publicly available resources that offer information on legislative hearings, they lack robust functionalities (Ruprechter, 2018). As a result, there is a need to format the data so it can be ingested and understood in a consistent, reproducible, and searchable way.

Digital Democracy, an initiative backed by the nonprofit, nonpartisan newsroom CalMatters, aims to increase transparency and accountability in the California legislature by making state government data accessible. The project offers simple, nonpartisan resources and a searchable database for journalists and the general public to learn about the legislature and gain insight into important government decisions. The hope is that increasing the public’s access to information about state government may help influence the legislature, foster civic engagement, and provide newsworthy content for journalists. This ambitious project is further summarized in the next chapter.

1.1 Digital Democracy

Digital Democracy started as an initiative founded by California Polytechnic State University’s Institute of Advanced Technology and Public Policy (IATPP) in 2014. As Dave Lesher, Co-Founder of CalMatters and the current director of the Digital Democracy initiative, outlines, the initiative as it currently stands seeks to:

1. Change the California legislature by increasing transparency and accountability.

2. Change civic engagement by offering the public easily accessible resources to learn about state legislators and issues.
3. Change journalism by offering new data and content that reveal government decision-making (Lesher, 2023).

The project’s history and architecture are explained in further sections.

1.1.1 Project History

When the Digital Democracy project was founded in 2014, its initiative was to increase transparency in state government by offering free, user-friendly resources for students, researchers, and the general public to monitor state legislative proceedings and other government affairs. Sam Blakeslee, founding Director of the IATTP and former California State Senator, championed the initiative, saying “We developed Digital Democracy to open up government [...] With this powerful new platform, Californians will be able to see exactly what people are saying as state laws are being written” (Cal Poly News, 2015).

The Digital Democracy project was in active development from 2014 to 2020, with its first website made publicly available in 2015 (Ruprechter, 2018). From 2014 to 2020, many students, researchers, and technologists contributed to the project. These contributions resulted in a centralized database, several web applications, and many software processes and scripts distributed across roughly 40 code repositories. While the Digital Democracy initiative initially focused on California, it became clear that the project was applicable to other states. Therefore, in 2017, New York was added to the tool and, in 2018, searchable information from Florida and Texas was added (Robertson, 2018).

In 2020, due to limited funding, the project was suspended and active development of the services ceased. In 2021, Cal Poly and the IATPP received interest from the
nonprofit news and media organization CalMatters to resurrect the Digital Democracy initiative and build an entire new suite of resources for journalists and the general public. In November of that year, with support from the Knight Foundation, Arnold Ventures, and the Lodestar Foundation, Cal Poly received funding from CalMatters to resurrect Digital Democracy and begin development on the new resources. Since CalMatters involvement and the resurrection of the tool, California is the only state currently represented; however CalMatters has expressed interest in duplicating the tool in other states around the country (Lesher, 2023). The newest iteration of the Digital Democracy initiative will go public in April 2024.

1.1.2 Project Architecture

The Digital Democracy project encompasses many software systems and resources. These resources are responsible for all aspects of the initiative, from ingesting and storing legislative data to generating and serving content and running front end web applications. One of the project’s critical platforms, and the primary focus of this thesis, is known as the Transcription Tool. The Transcription Tool is a software application designed for viewing and editing legislative hearing data. Specifically, the tool is responsible for the transcription and annotation of legislative hearings, in addition to a host of other important responsibilities. Figure 1.1 provides an overview of the Digital Democracy resources as it stands in early 2024. The boxes represent the initiative’s primary software stacks. This diagram was created by IATTP Research Director Foaad Khoosmood in late 2023.
Starting from the top left of the diagram, the Transcription Tool Video Downloader ingests legislative video sources and other data from state government websites. The Transcription Tool is then responsible for downloading and transcribing legislative hearing videos and associated information. It consists of both a user-facing application and many back-end software processes to ingest, store, and retrieve data. The tool is described in further detail in the next chapter. Below the Transcription Tool in the diagram is the Continuous Acquisition Module (CAM). CAM is responsible for importing and aggregating text, images, and other data from state government websites, such as the names of legislators, lobbyists, and organizations and amounts of gifts and donations made to legislators. CAM is a series of software processes to ingest and clean data. Both the Transcription Tool and CAM write data to a
database called the Digital Democracy Database (DDDB). This database, as well as the project’s two other data storage systems, are represented as cylinders on the diagram. The DDDB is a relational database, meaning its data is stored as rows and columns. In addition to storing data in the DDDB, the Transcription Tool downloads videos and stores them on an object storage service. Tip Sheets, in the top right of the diagram, pulls data from the DDDB and the object storage to generate phenoms, or meaningful associations, from the data to identify potential news stories and content for journalists. Tip Sheets uses artificial intelligence and other methods to generate this data, which is served on a front-end website. The data from the DDDB and TipSheets is ingested into another database, a graph database that stores nodes and relationships, for fast and efficient querying. This database is responsible for serving the CalMatters’ Digital Democracy website.

The initiative has three front-end software applications that can be accessed via the web. These include the Transcription Tool, the Tip Sheets portal, and the Digital Democracy website. While the Transcription Tool is only available to a limited number of users, the other resources are available to larger audiences. The Tip Sheets portal is accessible by CalMatters staff and journalists to view and discover potential newsworthy content, while the Digital Democracy website is available to reporters as well as the general public. The website displays searchable legislative information, including hearing, bill, committee, organization, and financial data. It also summarizes key legislative issues and salient topics in the state of California.

1.2 Methodology

This thesis improves the Transcription Tool and its associated processes by integrating new software, Natural Language Processing (NLP) techniques, and data and user
interface (UI) changes in an attempt to decrease the time it takes users to annotate and edit legislative hearing transcripts. Table 1 shows the main contributions of this paper and the relevant software and evaluation metrics used.

Table 1.1: Main Thesis Contributions

<table>
<thead>
<tr>
<th>Contribution</th>
<th>Software</th>
<th>Evaluation Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integration of speech-to-text service</td>
<td>AssemblyAI</td>
<td>Transcription accuracy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Word Error Rate, cosine similarity)</td>
</tr>
<tr>
<td>Integration of speaker diarization service</td>
<td>Pyannote</td>
<td>Speaker count</td>
</tr>
<tr>
<td>Speaker identification process and known-person matching</td>
<td>Question-Answer Model</td>
<td>Accuracy, precision, recall, F1 score</td>
</tr>
<tr>
<td>Text post-processing, UI, and data improvements</td>
<td>Java, Python</td>
<td>Transcription Tool logs</td>
</tr>
</tbody>
</table>

As part of this thesis, we introduced a new Automatic Speech Transcription (AST) service into the Transcription Tool to automatically transcribe the legislative hearings. To determine the service to integrate, we compared the transcription accuracy, cost, and model runtime of four separate AST services. Second, we integrated a new speaker diarization service into the pipeline. We evaluated potential diarization services against the tool’s system, using legislative hearings previously tagged as ground truth.

Prior to the resurrection of the tool, there was already a robust process in place to correct text misspellings and formatting errors (Ruprechter, 2018). We improved upon this process by adding more phrases and making use of additional parameters in
the AST model. Additionally, we integrated a new text-based speaker identification process into the tool that uses Named Entity Recognition (NER) and a Question-Answering machine learning model. We compared six NER libraries to evaluate model accuracy and speed. In addition, two separate speaker identification systems were evaluated.

From 2023 to 2024, there were many UI changes made to the tool to improve user experience. Only the changes relevant to improving the transcription editing process are mentioned in this thesis. Similarly, there were many data and database changes made throughout the writing of this thesis. Only those related directly to the transcription editing process are mentioned. The effectiveness of the relevant changes above is evaluated using a data collection and logging process previously existing in the tool (Ruprechter, 2018). Finally, we evaluate the overall cost improvement across all the newly integrated features using metrics retrieved from the Digital Democracy database. Specifically, we compare the average transcription editing times at different stages of development throughout this thesis.

1.2.1 Results

We evaluate the accuracy and runtime of several AST services, ultimately deciding to integrate a service known as AssemblyAI (AssemblyAI, 2024) into the tool. We evaluate an AssemblyAI transcript success rate of 99.4% after integration.

We evaluated three different speaker diarization tools, determining that the software Pyannote (Bredin, 2020) had the highest accuracy and fastest runtime. After Pyannote’s integration into the tool, we found an average percent difference of 19.65% between actual and predicted speakers. We compare two speaker identification methods, a rule-based approach and a Question-Answer model, determining that the second
method had the higher accuracy (95.6%). After the model was integrated into the tool, we evaluated a speaker identification success rate of 40.5%. Finally, we measured overall transcriber efficiency throughout this thesis, determining an improvement of 45.05% from the baseline to final version of the tool. We discuss factors contributing to this improvement in the Total Cost Improvement across All Features subsection of the Results.

1.2.2 Organization

This thesis is separated into seven chapters. Chapter 2, “Background,” provides background on relevant software and natural language processing terms and the metrics used for evaluation. Chapter 3, “Related Work,” discusses prior work done on the Transcription Tool, other transcription editing tools, and relevant contributions in the field of natural language processing, including automatic speech recognition, named entity recognition, speaker diarization, and post-processing techniques. Chapter 4, “Methodology,” discusses how the research was conducted, including the software implementations, algorithms, and data used for evaluation. Chapter 5, “Results,” reviews the evaluation metrics and outcomes of this research. Chapter 6, “Conclusion,” summarizes the work achieved by this thesis. Finally, Chapter 7, “Future Work,” considers the state of natural language processing and future improvements to the Transcription Tool.
Chapter 2

BACKGROUND

The work of this thesis lies at the intersection of computer science, data science, computational linguistics, and journalism. Consequently, there are many terms relevant to these fields included in this thesis. Below is an explanation of some common NLP terms and evaluation metrics. In addition, included is a description of the Transcription Tool and its key features. The tool has been in development for nearly a decade and, as a consequence, relies on preexisting software and infrastructure. Some of the specific software used is mentioned below.

2.1 Transcription Tool

The primary purpose of the Transcription Tool is to facilitate viewing and editing transcripts from legislative hearings. The tool contains two user types: an editor and an administrator. Editors are assigned tasks, or portions of transcripts to annotate, by administrators. The edit transcript page contains rows of utterances and an associated video player. An example transcription editing page is available in Figure 2.1. As the video plays, the tool advances rows, highlighting the current spoken utterance. Editors play the video and annotate the transcript by editing the start time, end time, or text and cutting or merging the utterances as needed using the buttons to the right of the utterance. In addition, editors are tasked with assigning the speaker of the utterance. These speakers are either known legislators, lobbyists, or other representatives or members of the general public. If the speaker is not in our existing database, the user has the ability to create a new speaker. The dropdown to the right
of the utterance allows editors to assign an alignment (for, opposed, neutral, etc) for the speaker. Editors can assign speakers to utterances in bulk, using the “Replace person” and “Align person with tag” buttons to the top right of the utterances.

An administrator user contains all of the functionality of an editor plus additional manager capabilities. Administrators are able to view all of the legislative hearings, edit their associated information and metadata, and send videos for transcription. In addition, they are responsible for creating and assigning tasks to users. They can also view summary statistics and metrics on the number of video hours, hours spent editing, and transcriber-specific information. An example administrator page is shown in Figure 2.2. This view shows a list of hearings, their House (Senate or Assembly), date, and associated video cuts in green. Each cut represents one segment of a hearing, running about 10 to 50 minutes in length. Summary information above the

Figure 2.1: Transcription Tool Edit Transcript View
hearings shows the number and duration of videos. This is important information as the administrator needs to task work for editors, ensuring videos are transcribed in a timely manner but also relatively evenly distributed amongst editors. Administrators can also view information about the status of a video as it progresses through the pipeline, from downloading to cutting to tasking. While hearings download automatically from the state government website, a process described in greater detail in the Methodology section, an administrator does have the ability to download a hearing manually using the “Insert Hearing Manually” button in the top left.

Most of the contributions in this thesis improve upon the automatic video transcription process prior to users viewing the transcripts. Some UI and data features were added to improve both the editor and administrator experiences.

![Image](image.png)

**Figure 2.2: Transcription Tool View Status of Hearing Administrator Page**
2.2 Software

There are many terms related to software development and computer programming referred to in this thesis. One such term is Application Programming Interface (API). An API is a standard way for computer systems to communicate. APIs are used to create, retrieve, or delete resources or data. Hypertext Transfer Protocol Secure (HTTPS) is a protocol used to transfer data between web browsers and servers.

Many of the Digital Democracy resources are hosted on Amazon Web Services (AWS). We employ Amazon’s Elastic Cloud Compute (EC2) to run many of the applications, as well as Simple Storage Service (S3) to store images, videos, and other data. The DDDB is a MySQL database hosted on Amazon’s Relational Database Service (RDS). Rekognition, Amazon’s computer vision platform, was previously used in the tool for facial recognition of speakers. Finally, the Transcription Tool uses crontab, AWS’s version of a cron job, or a background process that executes non-interfering jobs, to run software processes and database queries. At times throughout this thesis we refer to AWS and its suite of tools; these tools are what we are referring to.

The Transcription Tool front-end application is written in Java and uses the Java-based framework Spring Boot. It uses Apache Maven and Apache Tomcat for build automation and deployment. The other Transcription Tool back-end processes are written in Python and use the lightweight web framework Flask. The Transcription Tool is divided into several different code repositories. These and the other Digital Democracy repositories are hosted on GitHub.

We use many Python libraries for deriving evaluation metrics, including NumPy (Harris et al., 2020) and SciPy (Virtanen et al., 2020). In addition, we use the Python library JiWER (Vaessen et al., 2023) to evaluate some of the NLP systems.
We also rely on Python’s time module to measure the runtime of processes. We use displaCy, a software library and syntactical visualizer, to highlight words and explain core concepts.

Finally, this thesis refers to several different file types used in the transcription process. Some of those file types include text (TXT), SubRip Subtitle (SRT), Timed Text Markup Language (TTML) and JavaScript Object Notation (JSON). Common video file types include MPEG-4 Part 14 (MP4) and Waveform Audio File Format (WAV). A segmentation file (SEG) is a file type used in audio analysis to separate different types of sound into audio segments.

2.3 Natural Language Processing

NLP combines computational linguistics with statistics and machine learning to enable computers to understand text and speech. In NLP research, it is typical for unstructured text to be normalized via common pre-processing steps, such as tokenization, stemming, and lemmatization. Tokenization refers to the process of breaking a stream of unstructured text data into words, sentences, or symbols. After tokenization, the words, or tokens, are stemmed, wherein common suffixes or prefixes from a word are removed. Finally, the remaining stems are lemmatized to identify the inflected root of the word and return its base form.

In linguistics, a corpus refers to a collection of texts. These collections can be composed of a single language or many languages. In the Digital Democracy use case, the corpus is information related to legislative hearings, including bill versions, bill analysis, and legislative hearing transcripts. This thesis concentrates specifically on the committee transcripts. These transcripts consist of utterances.
is anything that a speaker says. In the Transcription Tool, utterances are displayed in the UI to be edited by users.

NLP systems use statistical language models to analyze text documents. One such model is known as “bag of words.” A “bag of words” model is used to analyze documents based on word count, discarding any information about word order or grammar.

Another term referred to in this thesis is a regular expression. Regular expressions are sequences of characters that specify a match pattern in text. They are represented as a special text string, and are useful for defining filters and for text search and replace.

2.4 Machine Learning

Machine learning models are core components of NLP research. One type of machine learning model used in this thesis is a question answering (QA) model. Question answering models retrieve answers to questions from a given text. Early examples of question answering systems include Baseball (Green Jr. et al., 1961) and LUNAR (Woods, 1973). These early systems were computer programs that answered questions about a specific domain using stored data. In the past several decades, QA models have become much more sophisticated, using machine learning and neural networks to determine responses.

The two primary types of QA models are known as extractive and abstractive. Extractive models determine the answer within the context provided. They typically result in concise, specific answers. Conversely, abstractive models generate a response by summarizing and synthesizing information from various sources. As each speaker’s name appears verbatim in the utterance, utilizing an extractive model fits this use case. This has the added advantage that extractive models are typically smaller and
faster. We discuss the Transcription Tool's implementation of a QA Model in the Methodology section.

### 2.5 Evaluation Metrics

Common evaluation metrics exist to measure the performance of NLP systems. Some of these metrics are used in the Results chapter and are useful to discuss. One common metric is known as Word Error Rate (WER). WER is derived from the Levenshstein distance formula, a measurement of the differences between two strings of text. Levenshstein distance is the minimal number of insertions, deletions, and substitutions of words or characters required to get two pieces of text to match. WER is the Levenshtein distance divided by the number of words in the text. The formula to compute the WER is:

\[
W_{ER}(\%) = \frac{S + I + D}{N} \quad (2.1)
\]

where \( S \) = substitutions, \( I \) = insertions, \( D \) = deletions, and \( N \) = number of words. The lower the WER value, the better the performance of the system.

A second metric used to determine the performance of a speech recognition system is Word Information Loss (WIL). WIL is the percentage of words incorrectly predicted between a set of ground truth sentences and a set of hypothesis sentences. The WIL formula is:

\[
W_{IL}(\%) = 1 - \frac{C}{N} + \frac{C}{P} \quad (2.2)
\]
where $C =$ number of correct words, $N =$ number of words, and $P =$ number of words in the prediction. Similarly to WER, the lower the value, the better performance of the system.

Cosine Similarity is yet another way to measure the similarity between two documents. When computing cosine similarity between text, words are represented as vectors and documents are represented in n-dimensional vector space. Cosine similarity is determined by taking the dot product and dividing it by the magnitudes of each vector. The equation of cosine similarity is:

$$\cos(a, b) = \frac{ab}{\|a\|\|b\|}$$

where $a \cdot b$ represents the dot product represents the dot product of vectors $a$ and $b$, $\|a\|$ represents the Euclidean norm (magnitude) of vector $a$, and $\|b\|$ represents the Euclidean norm (magnitude) of vector $b$. Cosine similarity is a value from 0 to 1, with 0 meaning the two vectors are orthogonal and entirely dissimilar. When two vectors are identical, the angle between them is 0, and the cosine similarity is 1. Cosine similarity assumes a "bag-of-words" vector representation, comparing unordered sets, whereas Levenshtein takes into account the order of the elements in the sequences.

Term frequency-inverse document frequency (TF-IDF) is a measure used in information retrieval and NLP research that quantifies the importance of a word or phrase to a document collection. The term frequency (TF) part of the equation represents the number of times a term appears in a document over the total number of words in the document. The inverse document frequency (IDF) part of the equation represents the proportion of documents in the corpus that contain that term. The formulas for both are:
\[ TF(i, j) = \frac{\text{Term } i \text{ frequency in document } j}{\text{Total words in document } j} \]  

(2.4)

\[ IDF(i) = \log_2 \frac{\text{Total documents}}{\text{documents with term } i} \]  

(2.5)

TF-IDF is calculated by multiplying the TF and IDF scores. The formula is:

\[ TF - IDF = TF(i, j) \times IDF(i) \]  

(2.6)

This metric is found in many NLP applications, including search engine ranking and text classification and summarization. TF-IDF is a value between 0 and 1. The higher the numerical weight value, the rarer the term; the smaller the weight, the more common the term.

Machine learning evaluation metrics also apply to NLP systems. For example, a confusion matrix is one common way to represent the outcomes in a classification model. A confusion matrix contains the four outcomes:

1. True Positive (TP). The model predicted True, and it should have predicted True.
2. False Positive (FP). The model predicted True, and it should have predicted False.
3. True Negative (TN). The model predicted False, and it should have predicted False.
4. False Negative (FN). The model predicted False, and it should have predicted True.
Several performance metrics can be calculated from the above outcomes, including Accuracy, Precision, Recall, and F1 Score. The formulas for these are:

\[
\text{Accuracy}(\%) = \frac{TP + TN}{TP + FN + FP + TN} \tag{2.7}
\]

\[
\text{Precision} = \frac{TP}{TP + FP} \tag{2.8}
\]

\[
\text{Recall} = \frac{TP}{TP + FN} \tag{2.9}
\]

\[
F_\beta = (1 + \beta^2) \times \frac{\text{Precision} \times \text{Recall}}{(\beta^2 \times \text{Precision}) + \text{Recall}} \tag{2.10}
\]

Precision, recall, and F1 score are bound between 0 and 1, with closer to 1 meaning higher performance. Accuracy is the degree of how close a measured value is to the actual value.

A Point-Biserial Correlation Coefficient is a measure of the strength of the association between a continuous variable and a binary variable. It is a special case of the more commonly used Pearson Correlation. We use the Point-Biserial Correlation in this thesis to determine the correlation between two variables, one with a binary outcome. The Point-Biserial correlation formula is:

\[
r_{pb} = \frac{M_1 - M_0}{\sigma} \sqrt{\frac{n_1 n_0}{N(N - 1)}} \tag{2.11}
\]

where \(r_{pb}\) is the point-biserial correlation coefficient, \(M_1\) and \(M_0\) are the means of the continuous variable for the categories where the binary variable equals 1 and 0, respectively, \(\sigma\) is the standard deviation of the continuous variable, \(n_1\) and \(n_0\) are the number of observations where the binary variable equals 1 and 0, respectively, and \(N\) is the total number of observations.
Lastly, the Diarization Error Rate (DER) is a common metric used to evaluate speaker diarization systems. The formula to derive DER is:

\[
DER = \frac{\text{False Alarm} + \text{Misses} + \text{Speaker Error}}{\text{Total Speech Time}} \times 100\% \quad (2.12)
\]

where false alarm is the total duration of segments wrongly assigned as non-speech, or speaker changes that do not correspond to actual changes, misses as the total duration of speech not detected, speaker error is the total duration of segments assigned to the wrong speaker, and total speech time is the total duration of the reference speech.
Chapter 3

RELATED WORK

There have been many improvements made to the Transcription Tool in the years since Digital Democracy was founded. The next sections elaborate on these contributions and discuss other existing transcription editing tools. In addition, this chapter discusses previous papers and areas of research relevant to this thesis and the field of NLP.

3.1 Other Transcription Editing Tools

While AST systems have achieved remarkable accuracy in recent years, many professional speech-to-text contexts still rely on humans to verify transcription accuracy (Blakeslee, 2015). Therefore, software tools and user interfaces that help users tag entities, create textual metadata, perform audio or video transcription, and edit subtitles and captions still exist both for commercial and research purposes (Ruprechter, 2018).

Entity tagging tools from Stenetorp et al. (2012) and Papazian et al. (2012) give users the ability to generate metadata and entity information from text (Ruprechter, 2018). Transcription editing tools give users the ability to view and correct text, often in association with a video or audio file. Software for transcription editing typically allows the user to play, pause, rewind, and further manipulate the audio or video files for additional clarity (Ruprechter, 2018). While NanoTrans (Seps, 2013) and ICS Caption Editor (Deshpande et al., 2014) are two research tools that allow users to create and edit transcripts, NowTranscribe Ltd (2017) and Cielo (2018)
are commercial products that offer similar functionality to the Digital Democracy’s Transcription Tool (Ruprechter, 2018).

3.2 Transcription Tool

The Transcription Tool has seen many contributions from professors, students, researchers, and other technologists since its creation. Notably, Rovin (2016) introduced improvements into the transcription process to increase efficiency, including speaker diarization, transcription user interface (UI) improvements, and text corrections. The diarization improvement merged consecutive utterances spoken by the same person. The major UI improvements included the addition of the ”Split Utterance” button, automatic assignment of fields for concurrent utterances, and functionality to import committee members into the tool’s speaker list. Validation was introduced that prevents transcribers from completing a task until specific criteria are met. Rovin also added an automatic text correction step that reformats bill numbers from their lexical representation into their equivalent numerical format, e.g. “SB”, to mean Senate Bill, “fifty-eight” would become “SB 58”. This text correction step is still used in the tool today. Rovin also added the capitalization of specific proper nouns.

Reinman (2016) improved the UI for the manager (admin) console, both for video cutting and adding meta information into the tool. The video editing page was simplified into two pages, one to manage the video cutting and the other for administrators to edit information for bills, tasks, committees, and hearings.

Lam (2016) implemented a video caching system called the Video Manager which sought to lessen the necessary disk space required for the tool to function to improve overall system performance. The caching system implemented used a Least Recently Used (LRU) cache, allowing for the videos to be stored on Amazon S3, a secure object
storage system (Amazon Web Services Inc., 2018) and loaded into the cache when needed.

In 2018, Ruprechter et al. introduced new features into the tool in four phases of UI and functionality improvements. The tool’s application framework was migrated from Stripes to Spring. The object-relational mapping approach utilizing OrmLite was removed and implemented using built-in Spring capabilities.

Another major effort undertaken by Ruprechter was the introduction of a data collection and logging system to quantify transcriber productivity and evaluate tool improvements. Client-side logs were gathered to measure user interactions with the tool, including editing utterances, changing speaker assignments, splitting and merging utterances, importing and searching for speakers, and video player interaction. A single user log would capture detailed information about the event. This information collected includes the name and time of the interaction, the user (transcriber) name, and the website element the user interacted with. The log information is collected and saved as JSON files on the server. A script runs nightly to save the information to a log table in the database. The data collection and user logging process was utilized in this thesis to measure software and UI improvements.

3.3 Natural Language Processing

Natural Language Processing (NLP) is a branch of computer science devoted to the understanding of human language. Lying at the intersection of computer science and linguistics, NLP encompasses both Natural Language Understanding (NLU) and Natural Language Generation (NLG). While NLU uses semantic and syntactic analysis of text to determine its meaning, NLG enables computers to generate intelligent responses. NLP systems often use machine learning and artificial intelligence techniques
to reveal the structure and meaning of text. NLP systems identify grammatical and syntactical patterns in text to derive its meaning using techniques like tokenization, stemming, lemmatization, and entity recognition. NLP has wide-ranging applications, including speech recognition, language translation, question-answering agents (e.g. chatbots), text summarization, and sentiment analysis. Specific NLP concepts relevant to this thesis are further defined in the following subsections.

3.3.1 Automatic Speech Recognition

Automatic Speech Recognition (ASR), also known as computer speech recognition or speech-to-text, is the capability of converting audio inputs into written text. The first attempt at ASR was developed at Bell Laboratories in the early 1950s. Called “Audrey”, it recognized only single digits (one to nine) spoken with known pauses by a single speaker (Davis et al., 1952). Another early attempt at ASR included IBM’s “Shoebox” machine, exhibited at the 1964 New York World’s Fair, which used audio pitches to identify one of 16 words (Aymen et al., 2011). In 1976, DARPA and Carnegie Mellon developed the Harpy System which used a heuristic search algorithm to identify over 1,000 words, or roughly the vocabulary of a three-year-old (Lowerre et al., 1976). The 1970s and 1980s also saw the introduction of traditional statistical approaches to ASR, including the Hidden Markov Model and the Dynamic Time Warp (Malik et al., 2021). The 1990s saw the use of stochastic language models and finite state machines to recognize human speech, including the more advanced use of syntax and semantics in ASR. Finally, the early 2000s saw the introduction of machine learning and concatenative synthesis to recognize an even larger vocabulary (Juang et al., 2005). The early 2000s also saw Gaussian Mixture Models (GMMs) used in combination with feed-forward artificial neural networks to further expand the capabilities of ASR. Finally, the past decade has seen the introduction of deep
learning and neural networks applied to ASR, such as Concurrent Neural Networks and Recurrent Neural Networks like long-short term memory (LSTM) models (Malik et al., 2021). Further recent advancements to speech processing include the use of attention mechanisms and transformers (Mehrish et al., 2023).

The field of ASR has wide-ranging applications, including caption generation, voice commands, dictation, message transcription, and intelligent voice assistants, particularly with the prevalence of Smartphones, video platforms, and social media applications. Many free, open source ASR systems exist today, including Kaldi ASR (Povey et al., 2011) and Julius (Lee et al., 2001) (Hernandez, 2023). While free, open source ASR models may be more flexible and allow for greater customization, commercial models, generally produced and maintained by large software companies, may have higher accuracy (Sciforce, 2021). Some popular commercial ASR systems include Google Cloud’s Speech-to-Text (Google, 2024), Amazon Transcribe (Leeper, 2020), IBM’s Watson (IBM, 2024), and Microsoft’s VALL-E (Wang et al., 2023). Smaller companies specializing solely in speech recognition and NLP have also entered the market, most notably Speechmatics (Speechmatics, 2024), Deepgram (Deepgram, 2024), AssemblyAI (AssemblyAI, 2024), and OpenAI’s Whisper (Radford et al., 2022). We include a comparison of several of these commercial speech-to-text systems later on in this work.

While ASR has improved remarkably from the 1950s with the introduction of the first speech recognition system to today’s deep learning approaches, challenges to the field still remain. Common ASR challenges include variability in speech, such as speed of talking and accents, vocabulary size and complexity, and the environment, such as background noises and overlapping conversations (Vadwala et al., 2017). Other difficulties in developing accurate, fast speech recognition include the need for large amounts of labeled data and the interpretability of models (Mehrish et al., 2023).
Finally, training AST models on languages for which there are limited resources or datasets poses a problem. There are over 6,000 languages in the world, and yet AST models are limited in the number of languages they support (Malik et al., 2021). Our use of AST for transcribing legislative proceedings is not immune to these challenges, and using an efficient, accurate speech-to-text service is highly important to the success of the Transcription Tool.

3.3.2 Named Entity Recognition

Named entity recognition (NER) is the process of locating and classifying designators from text into entity categories, such as organizations, locations, persons, etc (Li 2020). NER, a subtask of information extraction, is a preprocessing tool used frequently when performing text summarization, machine translation, information retrieval, and question answering. NER was first proposed at the The Sixth Message Understanding Conference (MUC-6) in 1996, which defined NER as “identifying the names of all the people, organizations, and geographic locations in a text.” (Grishman, 1996). Early NER approaches were rule-based, relying on dictionaries of terms or phrases based on existing vocabulary for a domain or entity type (Mansouri et al., 2008). In one of the earliest instances of NER, in the early 1990s, Rau used a rule-based approach to identify company names in financial news (Rau, 1991). Rule-based approaches generally use grammatical patterns, part-of-speech tagging, orthographic features, and dictionaries of domain-specific words or phrases to identify entities.

While rule-based NER strategies may work well for specific domains, they can be tedious to create, aren’t generalizable, and require continuous updating as domains and vocabularies grow (Mansouri et al., 2008). Due to these limitations, later NER methods rely much more heavily on statistics and machine learning, both unsupervised and supervised. One common unsupervised learning approach is clustering, which
relies on extracting named entities using context similarity. Feature-based supervised learning approaches rely on labeled data and feature engineering to train machine learning algorithms, such as Hidden Markov Models, Decision Trees, Support Vector Machines, and Conditional Random Fields. Finally, more recently, deep learning has been applied to NER, in the form of Convolutional Neural Networks, Recurrent Neural Networks, and Long-Short Time memory (LSTM) models (Li, 2020).

### 3.3.3 Speaker Diarization

Speaker diarization is the process of splitting up audio recordings that include a number of distinct speakers into homogeneous segments. It is essentially the task of determining ”who spoke when” in audio or video transcriptions (Anguera et al., 2012). Diarization typically consists of speaker segmentation, or determining where speaker changes occur, followed by associating segments of speech from the same speaker, or unsupervised identification (Tranter et al., 2006).

Initially proposed as an upstream processing step to ASR, diarization has evolved as a separate and distinct domain. The applications of diarization include, among others, speech and speaker indexing, document content structuring, speech translation, and speaker recognition (Anguera et al., 2012). Early applications of speaker diarization in the 1990s were for improving speech recognition in air traffic control dialogues and broadcast news recordings (Gish et al., 1991; Jain et al., 1996; Gauvain et al., 1998; Liu et al., 1999). Initial approaches to speaker diarization measured the distance between speech segments, and included generalized likelihood ratio (Gish et al., 1991) and Bayesian information criterion (Chen et al., 1998). The early 2000s saw diarization applied to other domains, including telephone and meeting conversations. Organizations encouraged the advancement of the field, such as the National Institute of Standards and Technology (NIST) (Park et al., 2022). This interest led
to new technologies such as beamforming (Anguera et al., 2007), and joint factor analysis (JFA) (Kenny et al., 2010). Later methods included i-Vector (Dehak et al., 2011), combined with principal component analysis (Shum et al., 2011), variational Bayesian Gaussian mixture model (Shum et al., 2013), and probabilistic linear discriminant analysis (Sell et al., 2014).

The Transcription Tool's use case for speaker diarization is twofold. First, accurate speaker diarization can help create better utterances, or blocks of speech spoken by a single speaker, during transcription. Second, accurate diarization has the potential to assist in speaker identification. When speaker changes occur, we can perform name extraction techniques in an attempt to identity the speaker. This process is elaborated upon in subsequent sections.

Many open source speaker diarization libraries have emerged in the last decade, from both research and commercial sources. These frameworks differ in their implementation language, hardware requirements, domains, and abilities for customization. SpeechBrain (Ravanelli et al., 2021), PyAudioAnalysis (Giannakopoulos, 2015), PyAnnote Audio (Bredin et al., 2023), FunASR (Gao et al., 2023), EEND (Fujita et al., 2019), and NVIDIA NeMO (Kuchaiev et al., 2019) are Python-based frameworks for speaker diarization. The Kaldi Speech Recognition Toolkit (Povey et al., 2011) uses Bash. LIUM_SpkDiarization (Meignier et al., 2010), a Java-based speaker diarization framework (2013), was used in a prior iteration of the Transcription Tool (Ruprechter, 2018). We provide a review of LIUM_SpkDiarization and PyAnnote Audio for the Transcription Tool use case later in this paper.

Despite significant advances in the field, challenges to speaker diarization still remain. Overlapping speakers and poor audio quality can impact the accuracy of diarization. A model trained on a particular domain may not apply well to a different use case,
known as domain mismatch (Park et al., 2022). Finally, integration with AST may prove difficult.
Chapter 4

METHODOLOGY

As mentioned, the Transcription Tool has been in development since the launch of the Digital Democracy project in 2014. The tool has had many contributors throughout the years, and thus is ever-evolving. A diagram of the Transcription Tool’s data pipeline as it looks in early 2024, created by longtime tool contributor Toshihiro Kuboi, is included in Figure 4.1. The tool’s primary processes are shown in distinct colors. The Video Downloader, represented by the blue boxes, retrieves and processes the legislative hearing videos into written transcripts that can be annotated by the tool’s users. The Video Downloader is one of the critical back-end services for the Transcription Tool.

Figure 4.1: Transcription Tool Data Pipeline (Kuboi, 2023)

The Transcription Tool front-end web application is represented by the pink boxes. The front-end includes the annotation UI, the Admin and Editor consoles, and various...
processes that generate user tasks and utterances. The Speaker Identification process, shown in purple, attempts to identify the utterance speaker using names extracted from the transcript. This process will be explained in greater detail in subsequent sections below. The Text Scripts process, shown in the orange boxes, handles the transcript post-processing and text corrections. Finally, the brown boxes represent the discussion segmentation and bill tagging process.

This thesis implements changes to several steps in the video processing pipeline, namely the transcription, speaker diarization, utterance post-processing, and speaker prediction, with the goal of increasing transcript accuracy and decreasing the time it takes editors to uplevel. The video downloading, cutting, trimming, and bill tagging processes remain the same from the previous to current version of the tool. Figure 4.1 shows the transcription process as an activity diagram, derived from an earlier activity diagram documenting the pipeline in the spring of 2018 (Ruprechtor, 2018). Processes are in light gray ovals, while external services are in rectangular boxes, such as the state government website. Result files written from the service are represented as document outputs.
The Transcription Tool changes were implemented from 2023 to 2024. While this thesis mostly changed aspects to the video processing pipeline, there were also UI and data changes made in the tool to improve editor experience and decrease transcript annotation time. Table 4.2 shows a summary of the new features and contributions as three different Transcription Tool versions: the features made prior to October 15, 2023 as Version 0, those made from October 15 to December 15, 2023, and those made from December 15, 2023 to March 10, 2023 (Versions 2 and 2.5). The subsequent sections of this methodology walk through the research and justification behind each new feature added to the tool.
Table 4.1: Transcription Tool Versions

<table>
<thead>
<tr>
<th>TT Version</th>
<th>Feature</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version 0</td>
<td>- Automatic transcription model</td>
<td>10/15/2023</td>
</tr>
<tr>
<td></td>
<td>- UI changes, end time propagation and shortcut fix</td>
<td></td>
</tr>
<tr>
<td>Version 1</td>
<td>- Automatically assign speaker ids for subsequent clips</td>
<td>12/15/2023</td>
</tr>
<tr>
<td></td>
<td>- Text post-processing improvements</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Addition of AST custom parameters</td>
<td></td>
</tr>
<tr>
<td>Version 2, 2.5</td>
<td>- Speaker diarization</td>
<td>2/29/2023</td>
</tr>
<tr>
<td></td>
<td>- Speaker identification</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Transformer Question-Answering model</td>
<td></td>
</tr>
</tbody>
</table>

4.1 Automatic Speech Recognition

One important step during the Transcription Tool pipeline is automatic transcription of the legislative hearings. Automatic transcription allows for a first-pass at transforming the speech from the videos into text. Automatic transcription occurs after the video cutting and trimming process. Video cutting ensures that videos are of a consistent length, running roughly 10 to 50 minutes in length, while video trimming ensures that long periods of silence, typically occurring at the beginning and end of legislative hearings, are removed (Ruprechter, 2018). After the hearing is trimmed and cut, it appears in the Admin interface of the Transcription Tool. Admins are able to make additional adjustments, deleting or manually trimming clips as needed.
Once the video cuts are reviewed, the Admin requests the video be sent for automatic transcription via the UI.

The Transcription Tool calls an API endpoint in the Video Downloader that sends the video to an external speech-to-text (STT) service. STT services use speech recognition to turn the speech from the video into written text. The video transcript response is then returned to the tool and saved in an output file that contains the full transcript of a single video, broken up into short segments of speech, otherwise known as utterances. The videos are stored on object storage in the cloud (AWS S3) and accessed through the Internet using HTTPS.

Cielo24 (Cielo24, 2018) was the first STT service used in the tool to generate automatic transcripts (Ruprechter, 2018). Cielo24 is a company providing automatic captioning and transcription. While the specifics of its pipeline aren’t documented publicly, its website states that it uses human-in-the-loop AI technology and guarantees a transcript turnaround time in two to twenty-four hours (Cielo24, 2018). Cielo24 was in use in the tool from 2014 to 2020. In 2018, Amazon Transcribe (Leeper, 2020) was added to the tool, offering users a second transcription service to use. Amazon Transcribe was chosen because of the project’s widespread use of other Amazon services. Users could select either service from a drop down in the UI.

Following the tool’s resurrection in the fall of 2022, we performed an analysis of current STT providers to help inform which STT service to integrate into the tool. With the advancement of deep learning and artificial intelligence in the field of NLP, faster, cheaper, and more accurate STT software may be available today compared to the services that existed when the tool was operational several years ago. The STT services we evaluated are outlined in the next section.
4.1.1 Speech-to-Text Services

STT services have improved dramatically over the last decade, especially with the rise of neural networks and deep learning. Therefore, we wanted to perform a quantitative assessment of available STT services to compare transcript accuracy, speed, and cost.

One important consideration was whether to choose an open or closed source STT service. Open source software is typically software that is stored in a public repository and distributed with its source code. Because its code is available publicly, open source software is often highly customizable. Other than hardware costs, there is typically no financial cost to use open source software. Despite these benefits, there are drawbacks to using open source tools. Development of open source software is typically unstructured, with informal testing and less robust quality assurance (Aberdour, 2007). There is also typically less documentation and project planning than proprietary software, as open source software often relies on volunteer developers contributing in their spare time. Software updates and bug fixes may be slower and ad hoc.

Of the popular open source STT services available today, we explored Project DeepSpeech (Hannun et al., 2014), Julius (Lee et al., 2001), Kaldi (Povey et al., 2011), OpenSeq2Seq (Kuchaiev et al., 2018), Flashlight ASR (Kahn et al., 2022), and OpenAI’s Whisper (Radford et al., 2022). Our basic requirements in choosing an STT service to evaluate were: 1) It is callable via Python. The video processing service is written entirely in Python, and it is preferred that we continue to use Python for consistency’s sake., and 2) It is actively maintained, as of the time of this research in late 2023. Of the available open source STT services we explored, several services - Julius, OpenSeq2Seq, and Project DeepSpeech - were last updated three years ago. In addition, Project DeepSpeech only supports transcription in English. Both Kaldi
and Flashlight ASR are written in C++ and have no built-in Python wrapper or interface. Therefore, of the available open source software packages we researched, we eliminated all but OpenAI’s Whisper, which we included in our evaluation.

The most expensive part of the Transcription Tool pipeline is the cost of the transcript upleveling (Ruprechter, 2018). The uplevelers, typically students, are paid hourly. Editing a transcript takes roughly three times the length of the video. That is, 10 minutes of video might take 30 minutes to edit. Low automatic transcription quality increases the time it takes editors to annotate a hearing. Conversely, the better the accuracy of the automatic transcription, the less time users have to spend editing. Compared to the amount paid to human editors, automatic transcription is cheap. It is typically billed at a constant rate, defined per minute of audio or video. An STT service that is free but has significantly worse accuracy would mean greater time and money spent on corrections.

There are many companies that offer some form of STT service. These services are proprietary; that is, their source code and implementation details are not available publicly. Some of these services are geared towards specific markets, such as smart phone technology, gaming, chatbots, or video analytics. Some specialize in certain areas of NLP, such as text summarization or intelligent agents (i.e. chatbots). Of the many closed source STT services we researched, we chose to evaluate Amazon Transcribe (Leeper, 2020), AssemblyAI (AssemblyAI, 2023), and Deepgram (Deepgram, 2023). All of these services have dozens of contributing developers, if not more. Given the small team of developers working on the Transcription Tool, offloading some of the work to an external provider may be beneficial.
4.1.2 Utterance Data

The data used to evaluate the services was utterances from 21 videos of California legislative hearings in 2018 running about 10 to 50 minutes in length. The total video length was 10:37:52. The data was utterance data previously annotated by users of the Transcription Tool. This data was sent to Loyola Marymount University Associate Professor of Law Michael Serota in 2022 for a different, unrelated research project. These utterances were first automatically transcribed using Cielo24, the tool’s previous STT service, and then subsequently by hand by the tool’s editors. The utterance metadata and their associated data types and descriptions are included in Table 4.2.

Table 4.2: STT Comparison Video Information

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
<th>Data Type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bill ID</td>
<td>The ID of the California State or Assembly bill</td>
<td>string</td>
<td>AB 1919</td>
</tr>
<tr>
<td>Hearing Date</td>
<td>The date of the hearing</td>
<td>date</td>
<td>8/31/18</td>
</tr>
<tr>
<td>Start</td>
<td>The start time of the utterance in seconds</td>
<td>integer</td>
<td>255</td>
</tr>
<tr>
<td>End</td>
<td>The end time of the utterance in seconds</td>
<td>integer</td>
<td>259</td>
</tr>
</tbody>
</table>
“Assembly Bill 1919 by Assembly Member Wood, an act relating to price gouging.”

The videos were first sorted by descending legislative year, starting with 2018, then by duration of the utterances (using the minimum start time and maximum end time for each video’s utterance). This resulted in a list sorted by the most recent videos of the longest length. The metadata and first utterance for the 10 longest videos can be found in Figure 4.3. A subset of the longest 21 videos was selected to test the STT services. To prepare the transcript from the videos as ground truth data, each video’s utterances were concatenated and written to an output TXT file.
It is important to note that the ground truth data still has errors and inconsistencies, as obtaining 100% accurate transcription data from a real-world setting is difficult. First, it is tedious and time-consuming for a human to manually transcribe speech. Second, the transcriber must make certain decisions when transcribing the data, and those decisions must remain consistent. For example, the transcriber must determine if certain words, such as filler words like “uh”, “um”, “oh”, and “hm”, should be transcribed, in addition to background noise or other voices. Transcribers must determine who takes precedence if there are overlapping speakers, and how to handle made-up words (whether to spell them phonetically, or as close to a real-world equiv-
alent as possible). Both humans and STT services make these decisions differently. Fortunately, the legislative hearings processed by the transcription tool are typically a primary speaker who speaks into a microphone. This amplifies the speaker’s voice above background noises.

4.1.3 Implementation

We used each service’s Python SDK, or equivalent Python client, to request transcription asynchronously for each video. Each service required some basic form of authentication, typically in the form of an API key or token. Each service’s implementation differed on the specifics but followed the same basic process outlined in Figure 4.4 below. The video data is loaded first, then a transcript is requested from each service. A process polls the service until the transcript is returned and saved. The transcripts are returned as JSON.

![Diagram for Transcription Request Process](image)

**Figure 4.4: Diagram for Transcription Request Process**
We transcribed each video in its entirety for each STT service. Each transcript response consisted of a list of utterance dictionaries with a start and end time, typically in milliseconds or seconds. A typical utterance dictionary might look like:

```json
{
    "text": "As a small business owner of two businesses [...]",
    "start": 339,
    "end": 349,
    "words": ["text": "As", "start": 339, "end": 340}, {...
}
```

Each transcript response is then saved to a TXT file using the minimum start and maximum end time of the ground truth data. This resulted in one output file for each video for each service, for a total of 84 files (21 videos x 4 services). In the Results section we provide a comparison of each service’s accuracy, runtime, and cost.

There was one implementation difference worth noting about Whisper in comparison with the other services. The other services give the ability to pass a URL to the video, while Whisper requires the raw bytes of the video for its transcription. This means that the videos must be downloaded prior to transcribing. Users of Whisper have requested that OpenAI provide a way to transcribe from URL, but as of March 2024 this functionality was still not implemented. We take this difference into consideration when determining what service to use.

### 4.2 Speaker Diarization

Another area we sought to improve in the tool was the transcript’s speaker diarization. Speaker diarization is the process of separating individual speakers in an audio recording in an attempt to answer the question “who spoke when?”. While ASR models tell us what is spoken, speaker diarization models tell us “who spoke it”.
In legislative hearings, the number of distinct speakers can be as little as two and as many as 100 (or more), especially when the hearing is open to the general public for comment. Public comment hearings often contain many new individuals who speak for short segments, making diarization difficult.

As of early 2024, there are few STT models with robust speaker diarization that come out-of-the-box. Of the STT models outlined in the previous section, both AssemblyAI and Amazon Transcribe can differentiate a maximum of 10 speakers. Deepgram does not publish its maximum number of speakers, but its documentation states, “We have seen very high accuracy of speaker identification on audio with 16+ speakers” (Doty 2022). Lastly, Whisper does not provide any speaker diarization.

The number of distinct speakers for a legislative hearing can be as little as two and as many as 100 (or more), we can’t rely solely on the AST speaker diarization for hearings with more than 10 (or, in the case of Deepgram, 16) distinct speakers. Therefore, we employ a separate speaker diarization model in our approach. If a legislative hearing has a low number of speakers detected, according to the AST model, we use the diarization returned from the model. If the model returns a count of the maximum number of speakers it can diarize, we run a separate diarization pipeline after the automatic transcription completes. The next section discusses the speaker diarization models available today for the case with 10 or more distinct speakers.

### 4.2.1 Diarization Services

There are fewer speaker diarization services than there are automatic transcription ones. The previous version of the Transcription Tool used the Java-based tool LIUM SpkDiarization, from the Laboratoire d’Informatique de l’Université du Mans (Meignier et al., 2010). While LIUM was used in the tool for several years, the soft-
ware was last updated four years ago and is no longer actively maintained. Therefore, we evaluated new speaker diarization services to replace LIUM.

One speaker diarization service on the market is NVIDIA’s NeMo (Kuchaiev et al., 2019), a conversational AI toolkit built for researchers working on NLP models. A second diarization service available is an open source Python toolkit called PyAnnote Audio (Bredin, 2023). PyAnnote was developed solely for the task of speaker diarization, while NeMo provides a generative AI framework for a variety of NLP tasks.

Mu, et al. (2023) compares the performance of NeMo and PyAnnote on two datasets, a simple dataset containing only two distinct speakers and a complex dataset containing a range, from one to 21 speakers, with an average of 6.5 distinct speakers. For the simple dataset case, the Diarization Error Rate of the models was similar (0.125 for PyAnnote and 0.118 for NeMo). For the complex dataset, NeMo’s DER was lower than PyAnnote (0.184 for NeMo and 0.342 for PyAnnote). NeMo’s higher performance doesn’t come without a cost, however, as its Real-Time Factor (a measure of the speed of the model) and it’s memory usage is much higher than that of PyAnnote. The study concludes that PyAnnote is the preferred tool if speed and memory are limitations. Conversely, if a lower DER is priority and time and memory are not constraints, NeMo should be chosen. In addition, NeMo performed better with noisy audio, audio that contained lots of overlapping speakers and background noise.

For the Transcription Tool use case, the audio is generally of high quality; the hearing is a formal setting in which participants speak one at a time into a microphone. In addition, the speed of the Video Downloader pipeline is a concern in the tool, and for the Digital Democracy initiative as a whole. These hearings are intended as sources for journalists and, as a consequence, we want to minimize the amount of
time between when a video is available on the state website and when it can be upleveled by editors. Due to this and the relatively clean nature of the audio, we chose to integrate PyAnnote as the speaker diarization service. We elaborate on the implementation specifics in the next section. In addition, we provide a comparison of the baseline service (LIUM SpkDiarization) to PyAnnote.

4.2.2 Diarization Data

To test PyAnnote, we use the same subset of 21 legislative hearing videos used for the comparison of STT services. To get the ground truth data, we query the database to retrieve the actual number of distinct speakers tagged for each video.

In addition, we compare the runtime and accuracy of PyAnnote against LIUM for one particularly complex video with over 100 distinct speakers. These comparisons can be found in the Results section.

4.2.3 Implementation

The speaker diarization is run as an asynchronous process after a video has been automatically transcribed. In addition to running a separate diarization process, we also enable speaker diarization in the automatic transcription request, so each utterance returned contains a speaker label. The rest of the speaker diarization process is shown in the activity diagram in Figure 4.5. The solid arrows represent the process flow; the dotted arrows represent when data is fetched or saved to either the database or an external output file.

First, the diarization pipeline fetches the undiarized videos from the database. The process determines the number of distinct speakers in the AST output file. As PyAn-
Note is time and resource intensive, especially when compared to the AST service, we avoid running the second speaker diarization model when possible. If the distinct speaker count is less than 10, we use the automatic transcription speaker labels and omit the PyAnnote process. If the speaker count is equal to or greater than 10, we assume the speaker count is greater than 10 and run PyAnnote. Once PyAnnote completes, the output is saved, and the AST output and diarization output are merged (the Merge Utterances step in the diagram, a process we discuss in greater detail in the next section). Then, the final transcript is saved and the speaker identification process begins. The final output is then fetched by the Transcription Tool and displayed in the front-end for users to annotate.

**Figure 4.5: Diagram of the Speaker Diarization Process**

This process was first run sequentially after the automatic transcription process; however, the speaker diarization model performance was found to be a bottleneck in the video pipeline. Consequently, later versions of the pipeline moved the process to
a cronjob to run every 15 minutes, allowing for parallelization of transcription and
diarization. Multiprocessing was also added in favor of multithreading, allowing the
process to utilize multiple CPUs to increase compute.

4.2.3.1 Merging Output

The output of PyAnnote is a list of dictionaries representing audio segments, each
containing a start time, end time, and speaker label. For example, one dictionary
might be \{“start”: 15, “end”: 30, “S01”\}, implying the model predicted that speaker
1 spoke from 15 to 30 seconds. The automatic transcription response from the STT
service is a similarly-formed JSON file, with each dictionary also containing a “text”
key that represents the predicted utterance spoken. After the speaker diarization has
ran, the diarization output and the transcription utterances must be combined. This
is done by determining the greatest overlapping time between both sets of start and
end times. The merging algorithm is as follows:

- Iterate over the transcription utterances
- Initialize utterance variables: start, end, text
- If (end-start) < 1.5s, assign it as an unidentified speaker; continue
- For each transcription utterance, iterate over the diarization list
- Initialize diarization variables: d_start, d_end, speaker, max_overlap, diariza-
tion_match
- if (d_start <= start and d_end <= end) or (d_start <= start and end <=
d_end) or (start <= d_start and d_end <= end) or (start <= d_start and end
<= d_end), calculate the overlap
The algorithm above finds the greatest overlapping diarization start and end time for every utterance start and end time to determine the utterance speaker. At the end of the merging process, an output dictionary is produced where each utterance contains a speaker label. These speaker labels are then stored in the database to be displayed in the UI for transcribers to annotate.

4.3 Speaker Identification

During transcription editing, users must identify the speaker of each utterance and assign a person’s name to that speaker label. These speakers are assembly members, lobbyists, other legislators, and members of the general public. If an editor is unable to determine the name of the speaker, the option to tag the speaker as unnamed is available.

This speaker identification step is one of the slowest tasks for editors. Ruprechter (2018) measured users’ average interaction ratios, or the time users spent performing certain tasks. The speaker identification step took 34.6% of the transcript’s total editing time. Therefore, this thesis implements an automatic identification processes in an attempt to improve the speaker identification step. Earlier versions of the tool utilized Rekognition, the cloud-based computer vision platform offered by Amazon, for face recognition of legislators and other hearing members. Due to its high cost and relatively poor performance, in late 2023, facial recognition was turned off in the tool and this alternative approach to speaker identification was proposed.
In closed committee hearings, hearing participants are limited to only the chairperson and committee members. These people are known representatives who speak during many legislative hearings throughout their two-year term. Because they are known representatives, lawmakers do not often introduce themselves when speaking. Conversely, open-session committee hearings are legislative hearings where members of the public are invited to testify. This gives members of the general public the opportunity to share their viewpoints about a particular bill. During this testimony, speakers often introduce themselves before stating their viewpoints. Consequently, the name of the speaker can be found in the text of the utterance. The speaker identification process attempts to recognize and surface this name to the transcriber. The new process for speaker identification, shown for an example utterance, is in Figure 4.6. The NER extraction and QA Model step will be elaborated on in upcoming sections.

![Figure 4.6: Diagram of the Speaker Identification Process](image)

Figure 4.6: Diagram of the Speaker Identification Process

The dashed boxes represent data, and the solid boxes represent processes. First, an utterance is passed to the NER model. The model tags any person entity types. These names are extracted and passed to a model to determine the speaker. This name is written to the database. The next several sections detail the steps to this process, including the NER and speaker identification models used.
4.3.1 Named Entity Recognition

As mentioned in Chapter 3, NER is a widely-used NLP method to locate and classify named entities in unstructured text (Li et al., 2020). NER categories typically include the names of people, organizations, locations, monetary values, dates and times, and quantities (Nadeau et al., 2007). Performing NER from speech has additional challenges compared to running NER on written text. Transcripts from STT services can result in misspellings and punctuation errors, which can make NER difficult (Nguyen et al., 2021). A misplaced comma or lowercase name can result in widely different NER model outcomes.

An example of NER tagging, using the Python packages SpaCy for tagging and DisplaCy for entity highlighting, is in Figure 4.7. The entity types are highlighted, with the person type in green. As shown, the model tagged the first person type (Alex Flores) correctly, but incorrectly tagged the second (Bill 1250).

![Figure 4.7: Example of NER Tagging using SpaCy](image)

While easy for humans to recognize that Bill 1250 is not a person, making this same distinction for machine learning systems is challenging. First, it is highly context dependent, i.e. the sentence, “My name is Bill Fourth” contains both a name and a number and is a valid person name. Furthermore, speech is highly variable; there are many ways in which individuals may self-identify. Finally, lexicons of names are ever-evolving, so models must be continuously updated to incorporate these changing trends.
In the Transcription Tool, NER is used after the asynchronous speaker diarization process. After the AST and diarization output have been merged, NER is performed on each utterance to identify the names of people. These names are then passed into a subsequent process to determine the individual speaker’s name, a process outlined in the next section.

The speakers in the legislative hearings include assembly members, legislative staff, lobbyists, and members of the general public, among others. While the transcript returned from the STT service is robust, grammatical mistakes and misspellings are still frequent. Therefore, we attempted to find an NER library that will handle well even in the case of transcription errors.

4.3.1.1 NER Utterance Data

The dataset used to test the NER implementations consists of 10,000 utterances spoken by legislators, lobbyists, and members of the general public during Senate and House California legislative assemblies from 2016-2020. Half of the utterances (5,000) are instances of speaker self-identification, wherein the text of each contains the speaker’s first and last name. They have been processed using both automatic machine transcription and human annotation. The human upleveling process includes correcting the text of the utterance and labeling the utterance with the name of the speaker, whether an assembly member, lobbyist, or member of the general public. Five of the rows with shortened utterances are included in Table 4.3.

Table 4.3: Sample of the NER Data

<table>
<thead>
<tr>
<th>First Name</th>
<th>Last Name</th>
<th>Text</th>
</tr>
</thead>
</table>
There are many implementations of NER models made available through popular, open-source NLP Python packages. The six packages we test for tagging person entities are: 1) SpaCy (its \texttt{en\_core\_web\_lg} model) (Honnibal et al., 2017), 2) Natural Language Toolkit (NLTK) (Loper et al., 2023), 3) BERT Base NER (Devlin et al., 2018) using Transformers, 4) Stanford NLP (Manning et al., 2014), and 5) Flair (Akbik et al., 2019).

The methodology for testing the NER packages follows the same basic steps:
1. Run the NER model against each utterance.

2. Extract the entity types from the tagged tokens, identifying between a PERSON and NON-PERSON.

3. Evaluate the PERSON tokens against the actual name identified.

Each package contains its own NER implementation function appropriate to the library (Steps 1-2), but the evaluation function used (Step 3) is the same for each library. In addition, we measure the average time for each package to run NER on a single utterance (Step 1). The outcomes in comparing the NER models can be found in the Results chapter.

4.3.2 Speaker Identification Approaches

Extracting the person names from the utterance text is the first step in the speaker identification process. The second step is determining which person from the tagged names is the speaker. This thesis tests two methods for the speaker identification. In late 2023, the speaker identification process used regular expressions. In early 2024, a more sophisticated approach in the form of an extractive QA machine learning model was integrated. The subsequent two sections outline both approaches.

4.3.2.1 Rule-Based Matching

The first approach for speaker identification used a token-based matching engine offered by SpaCy (Honnibal et al., 2017). Similar to regular expressions, the Token Matcher uses a rule-based approach to match text to a set of pre-defined patterns. As introductions in text often follow similar structures, such as “My name is” and “I
am”, using regular expressions as a first pass approach seemed like a reasonable way to determine the speaker name.

SpaCy’s matching engine accepts a list of phrases, where each phrase is a list of tokens represented as a dictionary. The key of the dictionary is the rule used to match. The value is determined based on the key. For example, the dictionary {“LOWER”: “hello”} specifies a token whose lowercase form matches “hello”, while the dictionary {“ENT_TYPE”: “PERSON”} represents a token with a person entity type. So this pattern would match the text “Hello Anna”, case insensitive. A selection of three of the patterns added to the matcher can be found in Figure 4.8.

```
patterns = [
    ["LOWER": "my"], ["LOWER": "name"], ["LOWER": "is"], ["ENT_TYPE": "PERSON"],
    ["LOWER": "i"], ["LOWER": "am"], ["ENT_TYPE": "PERSON"],
    ["LOWER": "i"], ["LOWER": "am"], ["TEXT": "Dr."], ["ENT_TYPE": "PERSON"],
]
```

**Figure 4.8: Three Token Patterns used for Speaker Identification**

The pattern implementation started with less than a dozen phrases; however, as speakers were failing to be identified in the tool, the number of phrases grew. In total, about 40 different phrases were added to the Transcription Tool’s matching engine. Initially, we proposed that there was a finite set of phrases individuals might use to introduce themselves. This assumption was quickly debunked, as we continued to add patterns to the matcher to incorporate specific use cases.

While token matching may be sufficient for some programs, it suffers from similar limitations as other rule-based systems. While rule-based systems are highly interpretable, they are inflexible. As the case with the speaker introduction token matcher, an exhaustive list of phrases must be included in the patterns for the engine to work.
well. For this reason, we explored other speaker identification approaches. Other limitations of the token matcher are elaborated on in the Results section.

4.3.2.2 Question-Answering Model

The second attempt at speaker identification utilized a supervised machine learning system in the form of an Extractive QA model (Craig et al., 2023). This model was built and submitted as a final project for Cal Poly’s class on Natural Language Processing (CSC 482).

The QA model added to the Transcription Tool uses Bidirectional Encoder Representations from Transformers, otherwise known as BERT, (Devlin et al., 2018), a language model based on the transformer architecture from researchers at Google AI Language. BERT applies the bi-directional training of the attention model Transformer to language modelling, showing that a bidirectionally trained language model performs better than a unidirectional language model. The model architecture is shown in Figure 4.9.

![Figure 4.9: BERT Model Architecture (Devin, 2018)](image-url)
The base model used for training was \texttt{bert-base-cased-squad2} (deepset, 2022). This model was previously trained on the Stanford Question-Answering Dataset (SQuAD), a reading comprehension dataset of questions posed by crowd workers on a set of Wikipedia articles. SQuAd2.0 adds an additional 50,000 unanswerable questions to the first version of the dataset (Rajpurkar et al., 2018).

Craig, et al. trained the QA model on a dataset of about 50,000 utterances with associated labels already annotated by the transcription tool. The label was either the full name of the speaker or, in the case of no named speaker, “Unidentified Speaker.” It is important to note that there are instances of speaker name misspellings in the training labels. In the case where only a first or last name is found, the code will try to determine the missing name by using a Levenshtein distance formula to determine if the preceding or subsequent word is misspelled. If the word is within the specified Levenshtein ratio when compared to the provided last name, the misspelled name is appended as the speaker label. Additionally, there are instances in the utterance where speakers introduce themselves with a nickname. For these cases, a custom dictionary is provided wherein if the name is found in the dictionary, both the name and its nicknames are added to the name extraction code. A sample of the nickname dictionary can be found in Figure 4.10.
In some cases, the middle names or initials of the speaker are in the utterance but not found in the label. In these cases, the labels were modified to include any middles names or initials found in the utterance. Finally, a number of utterances and their associated labels had mismatched hyphens (e.g. “Smith Rowe” is found in the utterance while the label is “Smith-Rowe”). For this case, a regular expression was added to ensure a match between the text and its label.

The training process consisted of partitioning the labeled dataset into a list of lists, with each sublist in the form [answer, utterance]. The data was shuffled and divided into a training and test set, with 80% for training and 20% reserved for testing. First, names are extracted from the dataset using tokenization and NER. The NER model used was SpaCy’s English transformer pipeline (the en_core_web_trf model).
This model has a named entity F-score of 0.90. Then, the dataset is reformatted into a list of dictionaries. Each dictionary takes the form:

```
{
    "question": "Of these speakers: <names>, who is speaking?",
    "answer": {
        "text": <answer>,
        "start": start index of the answer,
        "end": end index of the answer
    },
    "context": <utterance>
}
```

The list of dictionaries was then further divided into a training and validation dataset, 80% for training and 20% for validation. The dataset is then tokenized again to prepare the data inputs to the model, by performing text embeddings to convert the words to their vector representation. Craig et al. use the Python Transformers package (Wolf et al., 2020) to train and test their model. They also use the open-source library Aim to track validation loss and other evaluation metrics (Arakelyan et al., 2020).

During early training, the model reached a loss of 0.4 and failed to further improve. Craig et al. determined that the issue was the unidentified speakers. Because the model was extractive, it was failing to generate the phrase “Unidentified Speaker”, as it was not found in the utterance. The authors addressed this by appending the phrase “Unidentified Speaker” to the utterances. Once the “Unidentified Speaker” phrase was added, the validation loss fell to below 0.4. The validation loss curves for these experiments are shown in Figure 4.11.
The researchers trained the model on two scenarios. The first scenario included the names extracted from the NER process in the posed question. Figure 4.12 shows the validation loss curve for the final model with names included. The model’s lowest loss occurred at checkpoint 3400. After this point, the model loss increases as the model has begun to overfit.
The second scenario omitted the extracted names from the question posed. Instead of the question, “Of these people: <names>, who is speaking?” the question posed was simply “Who is speaking?”. The validation results from this model are shown in Figure 4.13. The lowest loss for this model occurred at checkpoint 3600.

![Figure 4.13: Validation Loss Curve for the Final Model with Names Omitted (Craig et al., 2023)](image)

### 4.3.2.3 Known Person Matching

The next step in the speaker identification process is matching the speakers identified to known people. After the speakers are identified from the QA model, they are written to a table called TT_SpeakerExtraction in the DDDB database. An asynchronous process queries this table and searches a second table, the Person table, for name matches. As middle names are uncommon in the Person database, we omit any middle names or initials from the query and only use first and last names. We save any matching people found into a table called TT_SpeakerPrediction.
The predicted speakers are then surfaced in the UI for editors to view. If there is no matching Person, an option to create the speaker will appear in the UI. If there is a matching person, the person’s first, last name, and image (if available) will appear. If there is no speaker predicted, the button is grayed out and deactivated in the UI. Figure 4.14 shows an example utterance in the UI that contains a speaker introduction.

In this case, the speaker identification magnifying glass is active and can be clicked. Figure 4.15 shows the modal window that appears when clicking the magnifying glass.

The model correctly predicted a speaker; however, the speaker’s last name is misspelled as “Chacha.” Because there was no found matching person in the database, the “Create new speaker” button appears. Clicking on this button opens the Person
dialog box, allowing the user to add a new person to the database. Figure 4.16 shows an example modal that appears when clicking to create a new speaker.

Figure 4.16: Create New Speaker Modal

As mentioned, the speaker identification feature was intended to help in open-session commitee hearings, in which members of the general public are invited to testify on a bill. Open commitee hearings contain new or lesser-known individuals who introduce themselves when speaking. Closed sessions are those that only include the legislators, and speaker introductions are less common.
4.4 Other Changes

We made relevant UI and data changes to improve the transcription editing process, detailed in the next three sections. It’s important to note that many other UI and data changes were made from 2023 to 2024 but are only indirectly related to reducing the transcription editing time and, because of this, are not mentioned in this thesis.

4.4.1 Text Post-Processing

A text post-processing step is included after the merge utterance step before the final TTML file is output. This post-processing step fixes common formatting and spelling errors found in the transcript. While STT services often have the ability for customization and fine-tuning, the models do have limitations. As is the case with the AST model integrated, it is unable to find and replace entire phrases, for example, as well as punctuation marks and other grammatical symbols. Consequently, we run a text post-processing step on the transcript after it is returned to fix common formatting and spelling errors.

An extensive number and bill formatting process was already in place from prior versions of the tool. This process changes the word version of a number to its nominal equivalent, i.e. “one thousand eighty five” to “1,085” (Ruprechter, 2018). The nominal form of digits is shorter and preferred in the transcript. Another common transcription mistake is the formatting of Assembly and Senate bills. The names of bills take the form of AB (Assembly Bill) or SB (Senate Bill) and typically two numbers, such as “AB 1423.” This is read as “A B fourteen twenty-three.” The transcripts returned from the AST service often incorrectly formats this number by adding
a space in between the numbers (i.e. “AB 14 23”). This post-processing step correctly reformats the bills before the final TTML file is written.

### 4.4.2 User Interface Changes

One significant UI addition to improve the human annotation time was introduced in the fall of 2023. As part of the transcription editing UI, editors have the ability to cut and merge utterances. The cut feature splits one utterance into two separate utterances, while the merge feature combines the current utterance with the utterance directly above. Both features are vital for the transcription editing process. When the cut utterance button is clicked in the UI, the utterance is split into two separate sections at the location of the cursor. The new utterance contains the same start and end time as the previous utterance. A typical editor flow is to cut the utterance then modify the end time of the previous utterance, as the time is shorter than it was before. A feature change, proposed by Transcription Tool administrator Hans Poschman, was to auto-populate the start time of the next utterance after the end time of the previous utterance was edited, immediately after a cut occurs. This allows the editor to avoid inputting both the end time of the previous utterance and the start time of the next utterance. Figures 4.17, 4.18, and 4.19 below represent this change.

![Figure 4.17: Utterance UI before the Cut Button is Selected](image_url)
Figure 4.18: Utterance UI after the Cut Button is Selected

Figure 4.19: Utterance UI after the End Time Edit

Figure 4.16 shows the utterance before it is cut. The cut button is in the right hand red circle. The start and end times are 0:00:01 and 0:00:35, respectively. Figure 4.17 shows the result after the utterance is cut. An additional row appears below the utterance that contains the text after the cursor. The start and end time for the new utterance matches that of the previous utterance (0:00:01 and 0:00:35). The editor then edits the end time of the previous utterance, shown in Figure 4.14. The new end time is auto-populated to be the start time of the next utterance. In the figure, both times are changed to 0:00:20. While this may seem like a trivial change, it prevents the editor from having to enter the new start time unnecessarily every time.
an utterance is cut. As cutting may occur dozens of times per transcript, this change avoids many redundant edits.

4.4.3 Speaker ID Propagation

Another change relevant to the transcription editing relates to automatically assigning people for subsequent tasks. One important editor responsibility is to assign a person, the speaker, for each utterance. At the beginning of a task, the utterances contain diarization labels, such as “S01” and “S02”, meaning Speaker 1 and Speaker 2. As transcripts are edited, users assign a person to each speaker label. These people are either new or pre-existing, in which case their name has already been saved to the Person table in the database. When an utterance is assigned a person, the speaker label is assigned a person ID, and the row is populated with the name of a person. An additional data change related to the propagation of these person IDs was proposed by Transcription Tool Administrator Hans Poschman.

It was proposed that subsequent unedited tasks from the same cut of video should be prepopulated to contain the diarization tag to person ID mapping from the preceding task. To give an example, imagine video A has two tasks (tasks 1 and 2). An editor finishes annotating Cut 1, and the diarization tag “S04” was assigned a person ID of 2908. This process takes all the prior assigned person IDs and populates the diarization IDs for subsequent tasks. So, the diarization tag “S04” is assigned the person ID 2908 for task 2. When the user begins task 2, the already assigned person IDs will be pre-assigned to the corresponding diarization tags and the names will appear next to the utterances. Updating the person IDs for subsequent tasks avoids the user from having to assign those same person IDs to the diarization tags. The process only runs for tasks in which editing has yet to start, as it was decided that changing speaker IDs when users are actively editing might make for a frustrating
user experience. The process runs every time an editor confirms the completion of a task in the UI. The full automatic assignment process is diagrammed in Figure 4.20.

![Diagram of the Process to Update Person IDs for Subsequent Tasks](image)

**Figure 4.20: Diagram of the Process to Update Person IDs for Subsequent Tasks**

The user completes Task 1 by clicking the Complete Task button in the UI. This triggers the update person ID (called pid in the diagram) process. The process gets the person IDs assigned for the diarization tags from Task 1 and updates the appropriate diarization tags with those person IDs for the unedited tasks. Only those tasks that have yet to be edited are updated. In the diagram, the editing process for task 2 has already started, and, as a result, it is not updated. Task 3 however has yet to be started and is updated. When a user begins task 3, utterances will have pre-assigned person IDs if the task contains the diarization tags from Task 1. To evaluate how effective this process is, we recorded the number of person IDs updated from January 30, 2024 to March 10, 2024. This metric is included in the Results section.
Chapter 5

RESULTS

This chapter discusses the results for each new model or service proposed as a new addition to the Transcription Tool. Where relevant, we include model or service comparisons. We also evaluate the new feature or tool after its integration into the tool when possible. Finally, we include the total cost improvement across all new features by analyzing the video and editing duration for each version of the tool.

5.1 Automatic Speech Recognition

In order to help determine the optimal ASR service to integrate into the video processing pipeline, we compared the accuracy, runtime, and a collection of other evaluation metrics for the STT services evaluated. The following sections describe the results from the comparison.

5.1.1 Comparison of ASR Services

The primary criteria we used to evaluate the ASR services were transcription accuracy, runtime in seconds, and service cost. Accuracy was evaluated using common speech recognition evaluation metrics, including WER, MER, WIL, and cosine similarity. In addition, the number of people and organization entities found using NER was compared to those found in the ground truth data. Customization options and tool usability were secondary qualitative criteria considered.
5.1.1.1 Transcription Accuracy

As mentioned in Chapter 2, WER, MER, and WIL are three common metrics used to evaluate transcription accuracy. Lower values indicate the ASR model is more accurate; higher values indicate lower accuracy. We derive these metrics by comparing the ground truth transcripts from our 21 videos to the transcripts produced by each service. The error rates are calculated using the Python package JiWER (Vaessen, 2023). Figure 5.1 shows the overall average WER, MER, and WIL for each service across the entire set of video transcripts. The total metrics by video for each service are available in Appendix A.

![Figure 5.1: WER, MER, and WIL for each Service](image)

Whisper has the lowest error rates, with a WER of 0.17, indicating that Whisper made the fewest mistakes in transcribing spoken words. Whisper’s MER is also the
lowest at 0.16. The metrics of the other three services are similar, hovering around 0.25.

We also calculated the cosine similarity of each hypothesis transcript against the ground truth. We measured cosine similarity using three different implementations from two Python libraries, SpaCy and Scikit-learn. Figure 5.2 shows the average cosine similarity metrics for each service for SpaCy, scikit-learn using CountVectorizer, and scikit-learn using TfidfVectorizer. CountVectorizer converts a text of documents to a matrix of token counts to produce sparse matrix representations, while TfidfVectorizer converts the text documents to a matrix of TF-IDF features.

![Figure 5.2: The Cosine Similarities Computed for each Service.](image)

Unlike the previous evaluation metrics, the higher the cosine similarity, the more similar the text is to the ground truth data. The cosine similarity metric when computed with SpaCy was similar for all of the services, greater than 0.99. When computed with scikit-learn however, there was more variability. AssemblyAI and
Whisper had the highest similarity metrics using both Scikit-learn implementations, followed by Deepgram and Amazon Transcribe.

5.1.1.2 Organization and Name Correctness

One important task of the transcribers is to correct the misspelled names of people and organizations. This can be a tedious process, as this may require editors to use external sources such as government datasets and other websites to ensure the correct spelling and identity of speakers. Many names have alternate spellings (e.g. Brittany and Britney) or sound much different than they are spelled (e.g. Siobhan). Therefore, evaluating the services by the number of names and organizations that match those found in the ground truth data may help minimize the number of manual edits required.

Using SpaCy’s `en_core_web_lg` Named Entity Recognizer, we tagged the person and organization entity types from the ground truth utterances. The NER model has a named entity F1 score of 0.85. The total number of distinct person entity types found in the ground truth data was 444, and the total number of distinct organization entity types was 372. We tagged the person and organization entity types across all of the transcripts for each service and compared them to those found in the ground truth data. Table 5.1 shows the number of matching person and organization entity types for the ground truth data compared with each service.
Table 5.1: Person and Organization Entities Found

<table>
<thead>
<tr>
<th>Service</th>
<th>People</th>
<th>People (%)</th>
<th>Orgs</th>
<th>Orgs (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ground Truth</td>
<td>444</td>
<td>NA</td>
<td>372</td>
<td>NA</td>
</tr>
<tr>
<td>AssemblyAI</td>
<td>213</td>
<td>47.97 %</td>
<td>107</td>
<td>28.76%</td>
</tr>
<tr>
<td>Amazon Transcribe</td>
<td>198</td>
<td>44.59%</td>
<td>152</td>
<td>40.86%</td>
</tr>
<tr>
<td>Deepgram</td>
<td>126</td>
<td>28.38 %</td>
<td>58</td>
<td>15.59%</td>
</tr>
<tr>
<td>Whisper</td>
<td>225</td>
<td>50.68 %</td>
<td>169</td>
<td>45.43%</td>
</tr>
</tbody>
</table>

Whisper performed the best, matching over 50% of the person entity types and over 45% of the organization entity types. AssemblyAI came in second place for matching person types, while Amazon Transcribe came in second place for matching organization entity types. The service that had the fewest matching person and organization entities was Deepgram.

5.1.1.3 Transcription Runtime

While transcription accuracy is highest priority, the speed of each service is also important. Choosing a fast service reduces the amount of time between the posting of the transcript on the state website and when it is ready to be annotated by editors. It is important that CalMatters reporters and the general public have access to the transcript and associated information in a timely manner, especially when the hearing is particularly newsworthy or salient. The previous transcription service used, Cielo24, guaranteed a turn around time of between 2 and 24 hours.
Using Python’s \texttt{time} module, we measured the time each service took to transcribe each video. The start time was set immediately before the transcript request was sent, while the end time was immediately after the response was returned. The runtimes were then aggregated across all the videos for each service. The results are displayed in Table 5.2.

Table 5.2: Runtime Comparison

<table>
<thead>
<tr>
<th>Service</th>
<th>Runtime (s)</th>
<th>Runtime (m)</th>
<th>Runtime as a % of Video Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon Transcribe</td>
<td>2809.82</td>
<td>0:46:49</td>
<td>7.45%</td>
</tr>
<tr>
<td>AssemblyAI</td>
<td>1060.50</td>
<td>0:17:41</td>
<td>2.93%</td>
</tr>
<tr>
<td>Deepgram</td>
<td>822.25</td>
<td>0:13:42</td>
<td>2.12%</td>
</tr>
<tr>
<td>Whisper</td>
<td>15100.59</td>
<td>4:11:40</td>
<td>38.88%</td>
</tr>
</tbody>
</table>

The combined length of all of the videos was 10:37:52. Deepgram had the fastest overall runtime, at 2.12\%, followed by AssemblyAI, Amazon Transcribe, and Whisper. Whisper’s runtime was significantly worse than the other services, over 5x as bad as the second slowest service, Amazon Transcribe. This may be due to our use of the open source version of the model.

Figure 5.3 shows the distribution of number of videos by transcription runtime for each service as a histogram. The bins represent the runtime as a percent of the video duration. For example, Deepgram transcribed 15 videos within 0-5\% of each video’s length. This demonstrates how uniform or distributed the transcription runtimes were for each service. AssemblyAI had the most videos falling in the first bin, with all 21
videos completing within 0-5% of the video length, followed by Deepgram, Amazon Transcribe, and Whisper.

![Histogram of transcription runtime as a % of video time.](image)

**Figure 5.3:** Histogram of transcription runtime as a % of video time.

### 5.1.1.4 Transcription Cost

While the automatic transcription cost is much lower than that of manual transcription, the price of each STT service is still an important factor. We will be running hundreds of hours of transcription every legislative session; an expensive service may be prohibitive to the transcription process and the Digital Democracy initiative at large. Table 5.3 shows each service’s cost of transcription per minute and the total cost summed across all videos. The cost per minute for each service was found in each service’s API documentation. The total cost was obtained by multiplying the cost per minute by the total video length.
<table>
<thead>
<tr>
<th>Service</th>
<th>Cost per minute</th>
<th>Total Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon Transcribe</td>
<td>$ 0.024</td>
<td>$ 15.31</td>
</tr>
<tr>
<td>AssemblyAI</td>
<td>$ 0.015</td>
<td>$ 9.57</td>
</tr>
<tr>
<td>Deepgram</td>
<td>$ 0.014</td>
<td>$ 9.19</td>
</tr>
<tr>
<td>Whisper</td>
<td>$ 0.00</td>
<td>$ 0.00</td>
</tr>
</tbody>
</table>

Whisper was the cheapest service, as we used the free version of the software. Deepgram was the second cheapest with a total cost of $9.19, followed by AssemblyAI ($9.57) and Amazon Transcribe ($15.31). The paid version of WhisperAI is $0.006 per minute, which would equate to about $3.82.

5.1.1.5 Sample Utterances

In addition to the quantitative evaluation metrics, we also compared a subset of the utterances returned from each service. While impossible to analyze all of the utterances, as there are hundreds, it is interesting to see the differences between services. Figure 5.4 shows a comparison of five utterances to the ground truth data.
This comparison reveals some interesting observations. For example, Amazon Transcribe captures many disfluencies that the other services do not. Deepgram transcribes the word version of numbers, such as “Sb eleven twenty nine” as opposed to “SB 1129”. It also fails to capitalize some organizations, such as the “Federal Bureau of Justice Statistics” in utterance 4. Whisper has some misspelling errors, such as “dissentrest” for “disinterest” in utterance 3 and “wave” for “waive” in utterance 5.
All of the services fail to transcribe the last sentence in utterance 5, “Okay, the floor is yours.” The audio may have been particularly low or garbled for this sentence. The models may also be unfamiliar with this expression, as it may not be common in conversational speech.

5.1.1.6 Selection of ASR Service

Ultimately, when comparing the service accuracy, cost, runtime, and utterances returned, we decided to integrate AssemblyAI’s model into the Transcription Tool. AssemblyAI provided a balance of accuracy, speed, and cost. Deepgram’s accuracy was lower than desired, while Amazon Transcribe was nearly a third more expensive. While Whisper had high accuracy and low cost, its slow runtime was prohibitive.

Subsequent sections discuss the integration of AssemblyAI into the tool. While this thesis only covers the integration of this service, we discuss the value in testing and integrating additional services in Chapter 7, “Future Work”.

5.1.2 AssemblyAI Custom Parameters

AssemblyAI allows for the addition of custom model parameters to improve the transcription accuracy. Two parameters that the Transcription Tool employs are the custom spelling and word boost parameters. The custom spelling parameter allows users to specify how certain words should be spelled or formatted. It is essentially a string find-and-replace. The parameter accepts a list of dictionaries where each dictionary has a “from” key and a “to” key. The “from” key is a single string or a list of strings that should be changed to the word found in the “to” key. The custom vocabulary list contains the names of cities and legislators that are frequently misspelled, as well as custom formatting for specific words and acronyms. The “from” key is case
sensitive but the “to” value is not. Some examples of word pairings present in the custom spelling list are “senator” to “Senator”, “SCIU” to “SEIU”, and “a.m.” to “A.M.”.

A second parameter we use is the word boost parameter. From the AssemblyAI Speech Recognition documentation, the parameter can be used to “boost certain words or phrases that appear frequently in your audio file” (AssemblyAI, 2024). The word boost list contains all of the currently sitting legislators names, as well as legalese commonly found in court or legislative hearings that the model might not otherwise be familiar with. This includes “aye” and “an act related to,” both of which the model struggles to get right. The request also accepts a boost parameter that specifies how much weight to apply to each keyword or phrase, one of “low”, “default”, or “high”. Specifying a high boost parameter may cause the model to overfit and change words unnecessarily, while specifying too low a parameter may prevent the model from correcting enough words. After some trial and error, we determined that the ideal boost parameter for the transcript requests was “low”. The addition of these parameters is free with the asynchronous transcription cost. The transcript request parameters as of March 2024 are shown in Figure 5.5.

```
transcript_request = {
    "audio_url": upload_url,
    "word_boost": word_boost,
    "boost_param": "low",
    "custom_spelling": custom_spelling,
    "speaker_labels": True,
    "punctuate": True,
    "format_text": True
}
```

Figure 5.5: The Transcript Request Parameters as of March 2024
The “speaker_labels” parameter specifies that speaker diarization should take place. The “punctuate” and “format_text” parameters specify that the transcript should include punctuation and text formatting.

5.1.3 AssemblyAI Metrics after Integration

AssemblyAI provides an online dashboard for its users to view metrics about their services, including metadata on the requested transcripts, errors returned, and spending activity. From April 2023 to March 2024, we have sent 4,451 total transcription requests. This also includes transcription testing prior to AssemblyAI’s integration into the tool. As of March 17, 2024, we have transcribed 2,109.82 hours of video using AssemblyAI for a total cost of $1,130.87.

Figure 5.6 shows the total transcription usage over time, from July 26, 2023 to March 17, 2024. Of the total transcript requests sent, we received 4,426 successful responses and 25 errors. This equates to a transcript success rate of 99.4%.
As seen in the figure, there was a spike in transcriptions in mid-January of 2024. The total number of transcriptions sent grew moderately from 2023 to 2024. Figure 5.7 shows the cost of transcription over time. Not surprisingly, it follows a similar pattern as the transcript usage over time.
5.2 Speaker Diarization

As discussed in the methodology, the Transcription Tool utilized LIUM SpkDiarization (Meignier et al., 2010) for its diarization process in its earlier implementation. We compared this software to the new speaker diarization model proposed. The next sections discuss the speaker diarization evaluation results.

5.2.1 Comparison of LIUM and PyAnnote

We compared the speaker diarization output and runtime of LIUM and PyAnnote for one legislative hearing video. The video tested has a length of 45:57 and contains a high number of speakers. It is hosted on Amazon S3 publicly and can be viewed at
https://shorturl.at/nL018. The video was transcribed by users of the Transcription Tool in late 2023. The video metadata is displayed in Table 5.4.

Table 5.4: Metadata for Comparison Video

<table>
<thead>
<tr>
<th>Video ID</th>
<th>44514</th>
</tr>
</thead>
<tbody>
<tr>
<td>File ID</td>
<td>bd3ded311505d363bdb52ce5ad699ba7</td>
</tr>
<tr>
<td>Duration (s)</td>
<td>2757</td>
</tr>
<tr>
<td>Legislative Session Year</td>
<td>2023</td>
</tr>
<tr>
<td>Hearing ID</td>
<td>256934</td>
</tr>
<tr>
<td>Hearing Subject</td>
<td>Assembly Labor and Employment Committee</td>
</tr>
<tr>
<td>Utterance Count</td>
<td>285</td>
</tr>
<tr>
<td>Distinct Person IDs</td>
<td>135</td>
</tr>
</tbody>
</table>

The output from both models is a list of audio segments, each containing a start time, end time, and speaker ID. LIUM outputs a file type called a segmentation file, with the file extension SEG. PyAnnote allows the user to write the output to any number of file types, but a TXT or JSON file are standard. Table 5.5 shows the speaker and segment counts from each service.

Table 5.5: Diarization Results for Test Video

<table>
<thead>
<tr>
<th>Service</th>
<th>Unique Speaker Count</th>
<th>Segment Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIUM</td>
<td>55</td>
<td>311</td>
</tr>
<tr>
<td>PyAnnote</td>
<td>107</td>
<td>306</td>
</tr>
</tbody>
</table>
As shown, the number of unique speakers predicted by PyAnnote (107 speakers) was closer to the correct number of unique people assigned for the video (135 people).

Table 5.5 shows the diarization runtime of each service. Both services were run on a 16GB 2021 MacBook Pro with 10-core CPU. LIUM’s runtime was longer (45 minutes) compared to PyAnnote’s runtime (23 minutes).

<table>
<thead>
<tr>
<th>Service</th>
<th>Runtime (m)</th>
<th>Runtime as a % of Video Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIUM</td>
<td>45.0</td>
<td>101.3 %</td>
</tr>
<tr>
<td>PyAnnote</td>
<td>23.4</td>
<td>49.5 %</td>
</tr>
</tbody>
</table>

The speaker count and runtime evaluation were two metrics we considered when assessing which diarization software to use. In addition to these, LIUM is also no longer actively maintained; it was last updated four years ago. PyAnnote is currently actively maintained, with a user base of over 4,000, as evidenced from the library’s follower count on GitHub.

5.2.2 Additional PyAnnote Metrics

There were several other tests we performed using only PyAnnote to measure the library’s performance. First, we diarized the same set of 21 videos used during the ASR service comparison. We compared the predicted number of distinct speakers to the actual number of person IDs tagged for each video in the Digital Democracy database. The results are in Figure 5.8. The total number of speakers predicted by
PyAnnote for the 21 videos was 720, while the total number of actual speakers labels was 905. The average percent difference between the predicted vs actual speakers is 23.03%. Typically, PyAnnote predicts fewer speakers than appear in the ground truth data. The complete percent difference table is included in Appendix B.

![Predicted vs. Actual Speaker Count for Videos](image)

**Figure 5.8: Predicted vs. Actual Speaker Count for Videos**

Lastly, while the code for the evaluations above was run locally, the diarization code for the Transcription Tool runs on an AWS EC2 instance. Determining an ideal instance size for the diarization to run efficiently but minimize cost is important. We experimented with different instance sizes to determine the impact of memory vs CPU. The same video from above (https://shorturl.at/nL018) was diarized for each test case. Table 5.7 shows the different instance sizes tested and results for each. Based on these results, we decided to go with a c4.xlarge instance that has 4 vCPU and 8GB of memory. It was slightly cheaper than the c5.xlarge instance but contains similar specs.
Table 5.7: Runtime of PyAnnote for Different AWS Instances

<table>
<thead>
<tr>
<th>Instance</th>
<th>vCPU</th>
<th>Memory (GB)</th>
<th>Diarization Runtime (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>m3.large</td>
<td>2</td>
<td>7.5</td>
<td>54.9</td>
</tr>
<tr>
<td>r4.large</td>
<td>2</td>
<td>15.25</td>
<td>29.0</td>
</tr>
<tr>
<td>c5.xlarge</td>
<td>4</td>
<td>7.5</td>
<td>23.0</td>
</tr>
<tr>
<td>d3.4xlarge</td>
<td>16</td>
<td>128</td>
<td>12.5</td>
</tr>
</tbody>
</table>

5.3 Speaker Identification

As described in the methodology, the speaker identification process runs after the speaker diarization. This process consists of tagging person entities using NER and updating the Digital Democracy database with the found names. As part of this research, we compared the speed and accuracy of six popular NER libraries. The results from this research are documented in the next section.

5.3.1 Named Entity Recognition

We compared the speed and accuracy of the six NER packages to determine which might be most beneficial to the speaker identification process. The evaluation metrics used were the F1 Score and runtime in seconds. These metrics are compared in the subsequent sections.
5.3.1.1 NER Accuracy

We chose to model the NER person tagging as a classification problem. Table 5.8 shows the available classification outcomes.

Table 5.8: Possible Classification Outcomes for NER Tagging

<table>
<thead>
<tr>
<th>Classification</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>True Positive</td>
<td>Name is correctly tagged as a person</td>
</tr>
<tr>
<td>True Negative</td>
<td>Name is correctly not tagged as a person</td>
</tr>
<tr>
<td>False Positive</td>
<td>Name is incorrectly tagged as a person</td>
</tr>
<tr>
<td>False Negative</td>
<td>Name is incorrectly not tagged as a person</td>
</tr>
</tbody>
</table>

These outcomes can be used to create a confusion matrix for each package.

Figure 5.9: Confusion Matrix for the Person Entity Tagging for SpaCy
Figure 5.10: Confusion Matrix for the Person Entity Tagging for Flair

Figure 5.11: Confusion Matrix for the Person Entity Tagging for BERT
From these matrices, we can derive accuracy, F1 score, and other metrics. Table 5.9 shows the derived metrics for each model.
Table 5.9: Evaluation Metrics for NER Models

<table>
<thead>
<tr>
<th>Model</th>
<th>Precision</th>
<th>Recall</th>
<th>Accuracy</th>
<th>F1 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flair</td>
<td>0.996</td>
<td>0.972</td>
<td>0.984</td>
<td>0.984</td>
</tr>
<tr>
<td>Stanford NLP</td>
<td>0.884</td>
<td>0.967</td>
<td>0.920</td>
<td>0.924</td>
</tr>
<tr>
<td>SpaCy</td>
<td>0.882</td>
<td>0.947</td>
<td>0.910</td>
<td>0.913</td>
</tr>
<tr>
<td>NLTK</td>
<td>0.872</td>
<td>0.879</td>
<td>0.875</td>
<td>0.875</td>
</tr>
<tr>
<td>BERT</td>
<td>0.901</td>
<td>0.752</td>
<td>0.835</td>
<td>0.820</td>
</tr>
</tbody>
</table>

Flair had the highest accuracy and F1 score, followed by Stanford NLP, SpaCy, NLTK, and BERT.

5.3.1.2 NER Speed

We measured the average time each NER model took to run on a single utterance. These averages are shown in Figure 5.14. The average NER runtime varied considerably. NLTK and SpaCy performed the best (0.1s), followed by BERT (0.09s), Flair (1.3s), and Stanford NLP (2.53s).
5.3.2 Speaker Identification Models

We tested two speaker identification methods in this thesis, as outlined in the methodology. The first used a rule-based approach, using patterns from common speaker introductions to extract the speaker names. The second used a QA machine learning model. The evaluation metrics for the two methods are discussed in the subsequent sections.

5.3.2.1 Rule-Based Matcher

As discussed prior, the name extraction approach first attempted was the token-based matching engine, called the Token Matcher, offered by SpaCy. The Token Matcher included over 40 phrases of speaker introductions, some as simple as “My name is
PERSON” or “I’m PERSON” to as complex as “PERSON, here on behalf of” and “Mr. PERSON, representing ORG”.

To evaluate the token matching engine against the QA model, we ran the matcher on the dataset of 47,000 utterances pulled from the Digital Democracy database (Craig et al., 2023). This was the same dataset used to train and test the QA model.

For both models, the speaker identification is treated as a classification problem. The possible outcomes are defined in Table 5.10.

<table>
<thead>
<tr>
<th>Classification</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>True Positive</td>
<td>Predicted speaker is correctly a name</td>
</tr>
<tr>
<td>True Negative</td>
<td>Predicted speaker is correctly unidentified</td>
</tr>
<tr>
<td>False Positive</td>
<td>Predicted speaker is incorrectly a name</td>
</tr>
<tr>
<td>False Negative</td>
<td>Predicted speaker is incorrectly unidentified</td>
</tr>
</tbody>
</table>

After shuffling the data, we ran each utterance through the Token Matcher and evaluated the results. The confusion matrix produced is shown in Figure 5.13.
We are able to derive evaluation metrics from the confusion matrix. These are reported in Table 5.11.

Table 5.11: Metrics for Token-Based Matcher, Full Dataset (47K Utterances)

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.8115</td>
</tr>
<tr>
<td>Precision</td>
<td>0.9075</td>
</tr>
<tr>
<td>Recall</td>
<td>0.6698</td>
</tr>
<tr>
<td>F1-Score</td>
<td>0.7707</td>
</tr>
</tbody>
</table>

We compare these metrics against those obtained from the best-performing QA model in the next section.
5.3.2.2 Question-Answer Model

As mentioned in the methodology, Craig et al. trained two separate QA models for the speaker identification task. The first included the speaker names in the question posed, while the second excluded the names from the question. The confusion matrices for both models are shown in Figure 5.16.

![Confusion Matrices Produced for the Final Model when Testing with Names Included and Names Excluded (Craig et al., 2023)](image)

**Figure 5.16: Confusion Matrices Produced for the Final Model when Testing with Names Included and Names Excluded (Craig et al., 2023)**

Each model’s accuracy, precision, recall, and F1-Score can be derived from the classification metrics. These are listed in Tables 5.12 and 5.13 below.

Table 5.12: Metrics for QA Model with Names Included

(Craig et al., 2023)

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.9226</td>
</tr>
<tr>
<td>Precision</td>
<td>0.8724</td>
</tr>
<tr>
<td>Recall</td>
<td>0.9864</td>
</tr>
<tr>
<td>F1-Score</td>
<td>0.9259</td>
</tr>
</tbody>
</table>
Table 5.13: Metrics for QA Model with Names Excluded
(Craig et al., 2023)

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.8818</td>
</tr>
<tr>
<td>Precision</td>
<td>0.8111</td>
</tr>
<tr>
<td>Recall</td>
<td>0.9880</td>
</tr>
<tr>
<td>F1-Score</td>
<td>0.8909</td>
</tr>
</tbody>
</table>

The results show that providing speaker names in the question posed does result in better performance. The confusion matrices show that in both scenarios, the model produces a similar number of true positives, but the model with included names has a lower false positive count.

Finally, the best performing model was tested using the full dataset of 47,000 utterances. The output confusion matrix is in Figure 5.17.
Figure 5.17: Confusion Matrix when the QA model is Run on the Full Dataset of 47K utterances.

Based on these values, we are able to derive the metrics found in Table 5.14.

Table 5.14: Metrics for QA Model, Full Dataset (47K Utterances) (Craig et al., 2023)

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.9561</td>
</tr>
<tr>
<td>Precision</td>
<td>0.9597</td>
</tr>
<tr>
<td>Recall</td>
<td>0.9875</td>
</tr>
<tr>
<td>F1-Score</td>
<td>0.9734</td>
</tr>
</tbody>
</table>
The metrics for the QA model are substantially better than those of the Token Matcher. The F1 score for the Token Matcher was 0.77, while that of the QA model was 0.97. For this reason, we integrated the QA model into the video pipeline in early 2024.

5.3.3 QA Model Results after Integration

As of mid-March 2024, there were 8,359 instances of speaker identification identified in the Transcription Tool using the QA model. We took a random sample of 100 rows and hand-checked them for accuracy. Of the 100 rows sampled, 69 were correct instances of speaker identification, 27 were incorrect in which the speaker identified was another name in the utterance but not that of the speaker, and 3 were unknown. This results in a success percentage of 69%. A sample of the speaker identification data is available in Figure 5.18. The first and last name of the speaker is included, as well as the confidence of the model and whether the model was correct, 1 for correct and 0 for incorrect. The utterance used by the speaker extraction model is also included. Utterances of the correctly identified speakers contain an introduction, such as “I'm Kristen ..” or “My name is Audrey..”.
The model confidence is a whole number between 0 and 100, with a higher number indicating greater confidence. We calculated the Point-Biserial Correlation Coefficient between the numerical column of confidences and the categorical column of correct or non-correct speaker. We calculated a correlation coefficient ratio of 0.583, indicating that the confidence value is only moderately indicative of whether the speaker identified is correct or not.

5.3.4 Other Changes

The other Transcription Tool changes made included a UI feature addition and the process to update person IDs for subsequent tasks. The UI change propagated the start time of the next utterance with the end time of the previous utterance, if an utterance was cut and the end time was modified. Unfortunately, the impact of this change is difficult to measure quantitatively, as there is no current mechanism set up in the tool to record the frequency of this scenario.
There are log metrics however to measure the impact of the process to update the person IDs for subsequent tasks. Every time a diarization and person ID is updated for a given video, a log statement is recorded for that person ID, diarization tag, and video ID. From January 30, 2024 to March 10, 2024, the time of this writing, 11,517 person IDs have been updated for their associated diarization tags for 901 distinct videos. If we assume an update time of five seconds per row, which is likely low, this has saved nearly 16 hours of transcription editing time.

5.4 Total Cost Improvement across All Features

To evaluate the total improvement to transcription editing time across all of the new features added to the Transcription Tool, we analyzed video and editing metrics across three separate tool versions, specifically the average transcription time in minutes per minute of task. This data came from the extensive user logging system put in place by Ruprechter in 2018. The data is show in Table 5.15.

Table 5.15: Comparison Metrics of Tool Versions

<table>
<thead>
<tr>
<th>Metric</th>
<th>Version 0</th>
<th>Version 1</th>
<th>Version 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version Dates</td>
<td>8/15/2023-10/15/2023</td>
<td>10/15/2023-12/15/2023</td>
<td>12/15/2023-2/15/2023</td>
</tr>
<tr>
<td>Number of Videos</td>
<td>618</td>
<td>706</td>
<td>1010</td>
</tr>
<tr>
<td>Number of Tasks</td>
<td>2876</td>
<td>2069</td>
<td>3083</td>
</tr>
<tr>
<td>Number of Editors</td>
<td>23</td>
<td>30</td>
<td>41</td>
</tr>
<tr>
<td>Total Editing Duration</td>
<td>838:30:49</td>
<td>986:50:02</td>
<td>1528:00:01</td>
</tr>
<tr>
<td>Average Transcription Time in Minutes per Minute of Task</td>
<td>3.37</td>
<td>3.28</td>
<td>2.31</td>
</tr>
<tr>
<td>---------------------------------------------------------</td>
<td>------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>Percent Ratio Improvement</td>
<td>NA</td>
<td>2.67%</td>
<td>34.70%</td>
</tr>
</tbody>
</table>

The versions were set in two month increments for fair comparison. As seen, the total video duration has more than doubled, from 314 hours to 702 hours, from Version 0 to Version 2 of the tool. This is understandable as the number of editors has also increased substantially. Surprisingly, the number of tasks decreased in Version 1 of the tool despite the number of videos having increased. The ratio of total video duration to total editing duration improved modestly from Version 0 to Version 1, with a 2.67% improvement. Surprisingly, the ratio improved significantly from Version 1 to Version 2 of the tool, showing a percent improvement of 34.70%. While Version 2 has a cut off date in the table of 2/15/2023, development continued from that date to the time of this writing. If the additional days are included in the metrics, from 2/25/2024-2/29/2024, the ratio continues to decrease, to 2.22, representing a percent change of 41.14%.

Scatter plots of the editing duration over video duration ratios aggregated and averaged over time for each version are shown in the figures below. Figure 5.19 shows the ratios for the baseline (Version 0) of the tool. There is a positive trend line, indicating a positive correlation between time and ratio.
Figure 5.19: Average Transcription Time in Minutes for Version 0

Figure 5.20 shows the average ratio over time for Version 1 of the tool. The trend line also has a positive slope, although it appears much more gradual.
Figure 5.20: Average Transcription Time in Minutes for Version 1

Finally, Figure 5.21 shows the average ratio over time for Versions 2 and 2.5 of the tool, from February 15, 2023 to March 10, 2023. This captures the new features up until the time of this writing. Interestingly, the trend line is slightly negative, indicating that, over time, the ratio is decreasing.
Finally, it may be useful to isolate the new transcribers for each version. This is to help rule out possible natural transcriber performance improvements as transcribers get more practice editing. Table 5.16 shows the count of new transcribers for each version of the tool. The unique transcribers from Versions 0, 1, and 2-2.5 were analyzed.

Table 5.16: New Transcriber Metrics across Versions

<table>
<thead>
<tr>
<th>Version</th>
<th>Number of New Transcribers</th>
<th>Average Ratio</th>
<th>Std Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version 0</td>
<td>10</td>
<td>3.86</td>
<td>4.53</td>
</tr>
<tr>
<td>Version 1</td>
<td>10</td>
<td>3.76</td>
<td>3.38</td>
</tr>
<tr>
<td>Version 2-2.5</td>
<td>17</td>
<td>2.72</td>
<td>2.12</td>
</tr>
</tbody>
</table>

**Figure 5.21: Average Transcription Time in Minutes for Versions 2 and 2.5**
Thirteen new transcribers started during Version 0 of the tool, 10 new transcribers started during version 1 of the tool, and 19 new transcribers started during version 2. As seen, the editing to video ratio for new transcribers still improved significantly, by 27.93%. Bar charts showing the ratio for the new transcribers for versions 0, 1, and 2-2.5 are shown in Figures 5.22, 5.23, and 5.24. The transcribers’ user names have been replaced by IDs to anonymize their identities (the X-axis).

![Bar chart showing the average editing time for new transcribers for Version 0.](image)

**Figure 5.22: Average Editing Time for New Transcribers for Version 0**
Figure 5.23: Average Editing Time for New Transcribers for Version 1

Figure 5.24: Average Editing Time for New Transcribers for Versions 2-2.5
The average editing duration to video duration ratio for new transcribers from version 0 of the tool was 3.86. The average ratio over all new transcribers from version 1 of the tool was 3.76, representing a 2.6% improvement from version 0. The average ratio of new transcribers from versions 2-2.5 of the tool was 2.72, representing a 27.7% improvement from version 1. From version 0 to version 2-2.5, the improvement was 29.5%.

Ultimately, while there were total cost improvements for versions 1 and 2 of the tool, it is important to not assume that the newly integrated features are the reasons for this improvement. While they may have contributed, there are other factors at play. For example, popular open session committee hearings, where many members of the general public come to testify, may be more frequent later in the year. As discussed prior, meetings with a high number of speakers are usually slower to annotate. In addition, we have continued to add new words and names to the ASR model parameters and post-processing steps. The AssemblyAI model also is in active development; the model used in Versions 1 and 2 of the tool may be better than earlier versions.
The Digital Democracy initiative is an ambitious project that will provide journalists and the public access to an enormous amount of information about the California state government. The initiative will offer a website and resources to journalists and users to learn about the legislative process in an attempt to increase transparency and accountability. These resources are provided by CalMatters and its funders free-of-charge.

This thesis evaluated and integrated a number of new NLP tools and techniques into the Transcription Tool software application in an attempt to increase transcript accuracy and decrease human annotation time. The next sections highlight our contributions.

6.1 Automatic Speech Transcription

We compared the accuracy, runtime, and cost of four AST services. Based on our analysis, we decided to integrate AssemblyAI into the tool, as it provided a balance of speed, accuracy, and cost. We included metrics on the total transcriptions since the addition of AssemblyAI, including the number of successful and failed attempts. The success rate of AssemblyAI in the Transcription Tool is 99.4%. 
6.2 Speaker Diarization

We compared the tool’s previous speaker diarization software, LIUM Spk_Diarization to a new diarization model, PyAnnote, determining that PyAnnote was faster and more accurate than LIUM. Second, we evaluated the number of speakers predicted by PyAnnote against groundtruth data. We discuss the integration of PyAnnote into the video processing pipeline, including the addition of a step to merge the transcription utterances with the diarization output.

6.3 Speaker Identification

In an effort to surface the names from speaker introductions, we tested two speaker identification implementations - a rule-based matching engine and a QA machine learning model. We determined that the QA model had a much higher accuracy (0.95) and F1 score (0.97) than the token matcher (0.81 and 0.77 respectively). We discuss the integration of the QA model into the system. Finally, we analyzed the results after integrating the model into the pipeline, determining that the model’s success rate was 40.5%. Approaches to increase the model’s success rate are included in the Future Work section.

6.4 Other Changes

The other relevant changes made to the tool include improvements to the text post-processing and the UI time propagation addition. In addition, we added a data change to update the person ID to speaker ID associations for subsequent tasks of the same video. We include a log metric to evaluate the effectiveness of this process.
6.5 Total Cost Improvement

We analyze the editing to video duration for all and new transcribers at three different stages of tool development. We determine that the latest version of the tool has a 37.32% efficiency improvement across all transcribers from version 0 to version 2 and a 35.01% efficiency improvement for new transcribers. While this improvement cannot necessarily be attributed solely to the new features, it is a promising result.
Our work to improve the Transcription Tool’s video processing pipeline has been successful; however, there is still work to be done to improve the accuracy and resiliency of the system. NLP is an ever-evolving field of research with models continuously advancing in performance and speed. In addition, the tool may benefit from other popular areas of NLP research, including multilingual language models, sentiment analysis, and text summarization. Advanced neural network architectures, such as Language Transformers and Transfer Learning, may benefit the tool in the future.

7.1 Improved Video Transcription Pipeline

Many of the contributions of this thesis attempted to improve the automatic transcription process. While we are proud of the work we have done, there are still areas of improvement. The next several sections propose new features or processes to further advance the video transcription pipeline.

7.1.1 Faster Speaker Diarization

The runtime for the speaker diarization process is a bottleneck in the automatic transcription pipeline. While improvements have been made, it is still typical for the diarization to take 40-50% of the video duration. If a robust proprietary speaker diarization software becomes available, it may be worth offloading this step. It may prove faster and more accurate, and worth the additional financial cost. As of spring
2024, AssemblyAI was in the process of improving the speaker diarization available in their model. Once this model can achieve a speaker count of several dozen, as opposed to just 10, we may be able to skip the separate diarization process entirely and simply use the labels returned in the automatic transcription response.

7.1.2 Additional ASR Services

The comparison of ASR services could be expanded to include additional services, such as Google’s Speech-to-Text (STT) or Whisper’s paid transcription plan. We evaluated four popular services in this thesis; other services could very well prove to have higher accuracy or run faster.

7.1.3 More Inclusive Speaker Identification

The speaker identification process relies largely on the correct spelling of names in the transcript returned from AssemblyAI. If an alternate or incorrect spelling is used, the names may not match any known people in the database. One area of work could be to generate alternate name spellings, i.e. “Eryn” or “Aaron” if “Erin” is returned, to maximize the number of potential matches. A set of commonly spelled names could be integrated, or the use of Levenshtein, to determine possible name matches. More than one name could be surfaced in the UI for the user to select.

7.1.4 Improved Fault Tolerance

Improvements could be undertaken to ensure the video processing pipeline continues to operate when a part of the system fails. For example, additional automatic transcription models could automatically run in the event of AssemblyAI failures.
The speaker diarization process has also errored or timed out in the past. While improvements have been made, the system should be resilient enough to ensure that processes continue and that manual intervention is not necessary.

7.1.5 Multilingual STT

Internationalization has also been a point of discussion. California is a diverse state and members of the general public may testify in other languages, such as Spanish and Hindi. Achieving high automatic transcription accuracy among languages other than English would give a more complete picture of the hearings. While AssemblyAI supports transcription for over twenty languages, it does not provide a way for multi-language recognition. A single transcript request must set a primary language, and the service does not yet have the capability to switch dominant languages mid-transcript. This would mean a separate language recognition model would have to run and segment the clips into distinct languages before sending them to AssemblyAI. As the multi-language use case appears fairly infrequently, a separate language recognition process may not be worth the performance cost at this time.

7.2 Automatic Data Refreshes

One of the parameters we include in the AssemblyAI request is a list of words, largely currently sitting legislator names and commonly misspelled acronyms. This data lives in the tool as a static file, hard-coded into the system. When a new legislative session begins, this custom vocabulary list will change, as legislators are added or removed. Exploring an automatic refresh process, where the names could be pulled and updated from an online database periodically instead of hard-coded into the service, may be beneficial. In addition, there has been some discussion about adapting the
tool to other states if successful in California. This would involve a large undertaking, particularly for other areas of Digital Democracy such as CAM and the Digital Democracy database. In the Transcription Tool, both the static data and scripts for accessing the state websites would need to be refreshed.

7.3 Faster Speaker Alignment

One of the tasks of the transcribers is to determine speaker alignment (for, against, neutral, etc). One aspect of work could be to predict speaker alignment via sentiment analysis. This would likely have to be a process that runs after speakers have been set, or else there is a risk that speaker utterances have been mislabeled and the prediction is running on utterances of different speakers.
BIBLIOGRAPHY


https://azure.microsoft.com/en-us/products/ai-services,

[64] Microsoft. Microsoft’s VALL-E.

https://www.microsoft.com/en-us/research/project/vall-e-x/,


## APPENDICES

### Appendix A

### EVALUATION OF EACH AST SERVICE PER TRANSCRIPT

<table>
<thead>
<tr>
<th>Video ID</th>
<th>AssemblyAI WER</th>
<th>AssemblyAI MER</th>
<th>AssemblyAI WIL</th>
<th>Amazon Transcribe WER</th>
<th>Amazon Transcribe MER</th>
<th>Amazon Transcribe WIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>126ef136cc26dace4447d59f1283583c</td>
<td>2272</td>
<td>2251</td>
<td>3435</td>
<td>2397</td>
<td>2206</td>
<td>3342</td>
</tr>
<tr>
<td>85e1f15aafa0a9b13e1f886b37e78b</td>
<td>1.920</td>
<td>1.892</td>
<td>2.915</td>
<td>2.253</td>
<td>2.053</td>
<td>3.005</td>
</tr>
<tr>
<td>095a33394788996f13e3e9f84c0e9f</td>
<td>2.381</td>
<td>2.362</td>
<td>3.551</td>
<td>1.946</td>
<td>1.888</td>
<td>3.155</td>
</tr>
<tr>
<td>699cccde79a9f9f13b2343d30ccf7f09b</td>
<td>2.746</td>
<td>2.708</td>
<td>4.157</td>
<td>2.116</td>
<td>2.020</td>
<td>3.298</td>
</tr>
<tr>
<td>117e7d75f5edbb101996c9762057e1</td>
<td>2.346</td>
<td>2.216</td>
<td>3.510</td>
<td>2.621</td>
<td>2.534</td>
<td>3.722</td>
</tr>
<tr>
<td>5e852f3d985c5c13e8a47c54692c01f</td>
<td>2.438</td>
<td>2.418</td>
<td>3.737</td>
<td>2.470</td>
<td>2.280</td>
<td>3.453</td>
</tr>
<tr>
<td>2b7c709f7e1c3390c88f6c0df57f8</td>
<td>2.504</td>
<td>2.497</td>
<td>3.809</td>
<td>2.183</td>
<td>2.118</td>
<td>3.480</td>
</tr>
<tr>
<td>31614cc9539e18986b0cc84f11336d005</td>
<td>2.518</td>
<td>2.490</td>
<td>3.819</td>
<td>2.467</td>
<td>2.334</td>
<td>3.701</td>
</tr>
<tr>
<td>4cb5898e7293c02aa7f522b97e9bb98</td>
<td>2.279</td>
<td>2.264</td>
<td>3.480</td>
<td>2.348</td>
<td>2.156</td>
<td>3.246</td>
</tr>
<tr>
<td>80660919fc792c3b89e17705c3a3ab</td>
<td>3.421</td>
<td>3.393</td>
<td>5.064</td>
<td>2.523</td>
<td>2.436</td>
<td>3.959</td>
</tr>
<tr>
<td>535b133111c3e3e107ce8264c8516efc</td>
<td>2.029</td>
<td>2.003</td>
<td>3.038</td>
<td>2.290</td>
<td>2.103</td>
<td>3.153</td>
</tr>
<tr>
<td>24b81e19229a173a868d5009e6ee50</td>
<td>2.557</td>
<td>2.516</td>
<td>3.709</td>
<td>2.256</td>
<td>2.087</td>
<td>3.167</td>
</tr>
<tr>
<td>e7e1b5a90edbe8c8977f27d2c94c572</td>
<td>2.681</td>
<td>2.648</td>
<td>3.990</td>
<td>2.188</td>
<td>2.072</td>
<td>3.254</td>
</tr>
<tr>
<td>f1a0c21ab0b0e648a8da27446f2c9965f</td>
<td>2.425</td>
<td>2.395</td>
<td>3.696</td>
<td>2.218</td>
<td>2.167</td>
<td>3.391</td>
</tr>
<tr>
<td>d073a3d434c23e0fe9a10ade592c9bd</td>
<td>2.421</td>
<td>2.399</td>
<td>3.564</td>
<td>3.129</td>
<td>2.727</td>
<td>3.843</td>
</tr>
<tr>
<td>539a8c47b9a2fa4f9f440c039767b63a</td>
<td>2.654</td>
<td>2.635</td>
<td>3.959</td>
<td>2.633</td>
<td>2.457</td>
<td>3.831</td>
</tr>
<tr>
<td>5c121d22784a861395e4a2b32018cad92</td>
<td>3.158</td>
<td>3.121</td>
<td>4.533</td>
<td>3.930</td>
<td>3.412</td>
<td>4.944</td>
</tr>
<tr>
<td>8e9b8beaca155ebe234a41aa46748c</td>
<td>2.400</td>
<td>2.378</td>
<td>3.642</td>
<td>2.558</td>
<td>2.323</td>
<td>3.465</td>
</tr>
<tr>
<td>915de9f8e1d70f7e2636da2e0350a0f5</td>
<td>2.402</td>
<td>2.393</td>
<td>3.866</td>
<td>1.676</td>
<td>1.642</td>
<td>2.811</td>
</tr>
<tr>
<td>2a7585a2ae060f1b3b0e0639b54ba2a</td>
<td>2.746</td>
<td>2.715</td>
<td>4.069</td>
<td>2.903</td>
<td>2.643</td>
<td>3.940</td>
</tr>
<tr>
<td>e1fbb08aac5ac5e1ac7b46d09a2fd22</td>
<td>3.495</td>
<td>2.233</td>
<td>3.945</td>
<td>3.312</td>
<td>2.041</td>
<td>3.312</td>
</tr>
</tbody>
</table>

Figure A.1: WER, MER, and WIL for AssemblyAI and Amazon Transcribe
<table>
<thead>
<tr>
<th>Video ID</th>
<th>Deepgram WER</th>
<th>Deepgram MER</th>
<th>Deepgram WIL</th>
<th>Whisper Al WER</th>
<th>Whisper Al MER</th>
<th>Whisper Al WIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>f29ef136cc2fdacece4447d5f123583c</td>
<td>2.400</td>
<td>.2275</td>
<td>.3578</td>
<td>1.631</td>
<td>1.579</td>
<td>2.590</td>
</tr>
<tr>
<td>856c15aaaf0a9b13e1fd86e307a976b</td>
<td>2.326</td>
<td>.2207</td>
<td>.3468</td>
<td>1.299</td>
<td>1.268</td>
<td>2.098</td>
</tr>
<tr>
<td>095a33f9473888696fe0c3e6984de09f</td>
<td>2.633</td>
<td>.2358</td>
<td>.4072</td>
<td>.1570</td>
<td>1.541</td>
<td>2.594</td>
</tr>
<tr>
<td>699c0e09a7df9f813b2343d3cc47f09b</td>
<td>2.786</td>
<td>.2656</td>
<td>.4213</td>
<td>.1759</td>
<td>.1724</td>
<td>2.882</td>
</tr>
<tr>
<td>f17e7d75c5edd8db01609c97620f7e1</td>
<td>2.914</td>
<td>.2730</td>
<td>.4162</td>
<td>.2525</td>
<td>.2331</td>
<td>.3506</td>
</tr>
<tr>
<td>f5e852f3d0f5c5c138a47c546920c0f1</td>
<td>2.544</td>
<td>.2442</td>
<td>.3983</td>
<td>.1748</td>
<td>.1697</td>
<td>2.764</td>
</tr>
<tr>
<td>237c0479cc7fa1c3390c86fbd057b8</td>
<td>2.379</td>
<td>.2316</td>
<td>.3758</td>
<td>.1384</td>
<td>.1370</td>
<td>2.328</td>
</tr>
<tr>
<td>3614cc9539e189600d0ec84f163e0605</td>
<td>2.625</td>
<td>.2504</td>
<td>.3939</td>
<td>.1760</td>
<td>.1723</td>
<td>2.885</td>
</tr>
<tr>
<td>4cb8858e672f3c2aa7f042b97e6bb9</td>
<td>2.505</td>
<td>.2374</td>
<td>.3652</td>
<td>.1463</td>
<td>.1424</td>
<td>2.355</td>
</tr>
<tr>
<td>80660f09af792c3b89e17f0f3e3a3ab</td>
<td>3.083</td>
<td>.2973</td>
<td>.4700</td>
<td>.1935</td>
<td>.1876</td>
<td>3.083</td>
</tr>
<tr>
<td>535b133111c3ea107ce828c48516efc</td>
<td>2.342</td>
<td>.2202</td>
<td>.3423</td>
<td>.1327</td>
<td>.1302</td>
<td>2.157</td>
</tr>
<tr>
<td>24b81e1029fa73a886df50093e6e2e50</td>
<td>2.654</td>
<td>.2510</td>
<td>.3885</td>
<td>.1681</td>
<td>.1637</td>
<td>2.706</td>
</tr>
<tr>
<td>e71eb5a00de8ce897f727d2c94c5f72</td>
<td>2.831</td>
<td>.2649</td>
<td>.4068</td>
<td>.1717</td>
<td>.1668</td>
<td>2.726</td>
</tr>
<tr>
<td>f1a0c21ab06e684a8ada2744e62c9065f</td>
<td>2.546</td>
<td>.2451</td>
<td>.3953</td>
<td>.1571</td>
<td>.1538</td>
<td>2.606</td>
</tr>
<tr>
<td>d07d3ed43d4c23e90ee9a1de592fbcd</td>
<td>2.614</td>
<td>.2480</td>
<td>.3853</td>
<td>.1744</td>
<td>.1703</td>
<td>2.795</td>
</tr>
<tr>
<td>539a8b347fb2a2fa4fe0c3967d030a</td>
<td>2.982</td>
<td>.2839</td>
<td>.4441</td>
<td>.1598</td>
<td>.1573</td>
<td>2.807</td>
</tr>
<tr>
<td>53c1211d227848e61395e4ba2018cad92</td>
<td>3.387</td>
<td>.3228</td>
<td>.4926</td>
<td>.2496</td>
<td>.2377</td>
<td>3.709</td>
</tr>
<tr>
<td>8e9f8eeaca15be23a41aa4a6748c</td>
<td>2.507</td>
<td>.2369</td>
<td>.3720</td>
<td>.1336</td>
<td>.1313</td>
<td>2.234</td>
</tr>
<tr>
<td>9158d99e3e707ef6e6386a23b035a0f5</td>
<td>2.030</td>
<td>.1984</td>
<td>.3278</td>
<td>.1265</td>
<td>.1246</td>
<td>2.169</td>
</tr>
<tr>
<td>2a75ba2ae6e1f13b0e6395b4bbaa2</td>
<td>2.772</td>
<td>.2659</td>
<td>.4176</td>
<td>.2090</td>
<td>.2028</td>
<td>3.317</td>
</tr>
<tr>
<td>e1fa08a0ac5ac01ac17b48d09a2f22</td>
<td>3.670</td>
<td>.2271</td>
<td>.3670</td>
<td>.2050</td>
<td>.1185</td>
<td>2.000</td>
</tr>
</tbody>
</table>

*Figure A.2: WER, MER, and WIL for Deepgram and Whisper*
Appendix B

ACTUAL VS PREDICTED SPEAKER COUNTS OF PYANNOTE ACROSS VIDEOS

<table>
<thead>
<tr>
<th>Cut ID</th>
<th>Predicted</th>
<th>Actual</th>
<th>Absolute Difference</th>
<th>Percent Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>30449</td>
<td>7</td>
<td>8</td>
<td>1</td>
<td>(1/8) * 100 = 12.5%</td>
</tr>
<tr>
<td>30450</td>
<td>83</td>
<td>128</td>
<td>45</td>
<td>(45/128) * 100 = 35.16%</td>
</tr>
<tr>
<td>30451</td>
<td>36</td>
<td>46</td>
<td>10</td>
<td>(10/46) * 100 = 21.74%</td>
</tr>
<tr>
<td>30452</td>
<td>20</td>
<td>27</td>
<td>7</td>
<td>(7/27) * 100 = 25.93%</td>
</tr>
<tr>
<td>30453</td>
<td>9</td>
<td>9</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>30454</td>
<td>9</td>
<td>11</td>
<td>2</td>
<td>(2/11) * 100 = 18.18%</td>
</tr>
<tr>
<td>30455</td>
<td>15</td>
<td>14</td>
<td>1</td>
<td>(1/14) * 100 = 7.14%</td>
</tr>
<tr>
<td>30456</td>
<td>40</td>
<td>64</td>
<td>24</td>
<td>(24/64) * 100 = 37.5%</td>
</tr>
<tr>
<td>31761</td>
<td>27</td>
<td>27</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>31762</td>
<td>43</td>
<td>40</td>
<td>3</td>
<td>(3/40) * 100 = 7.5%</td>
</tr>
<tr>
<td>31763</td>
<td>107</td>
<td>135</td>
<td>28</td>
<td>(28/135) * 100 = 20.74%</td>
</tr>
<tr>
<td>31764</td>
<td>9</td>
<td>8</td>
<td>1</td>
<td>(1/8) * 100 = 12.5%</td>
</tr>
<tr>
<td>31765</td>
<td>30</td>
<td>29</td>
<td>1</td>
<td>(1/29) * 100 = 3.45%</td>
</tr>
<tr>
<td>31827</td>
<td>106</td>
<td>123</td>
<td>17</td>
<td>(17/123) * 100 = 13.82%</td>
</tr>
<tr>
<td>31828</td>
<td>73</td>
<td>92</td>
<td>19</td>
<td>(19/92) * 100 = 20.65%</td>
</tr>
<tr>
<td>31829</td>
<td>20</td>
<td>26</td>
<td>6</td>
<td>(6/26) * 100 = 23.08%</td>
</tr>
<tr>
<td>31830</td>
<td>26</td>
<td>33</td>
<td>7</td>
<td>(7/33) * 100 = 21.21%</td>
</tr>
<tr>
<td>31831</td>
<td>24</td>
<td>25</td>
<td>1</td>
<td>(1/25) * 100 = 4%</td>
</tr>
<tr>
<td>31832</td>
<td>11</td>
<td>19</td>
<td>8</td>
<td>(8/19) * 100 = 42.11%</td>
</tr>
<tr>
<td>31833</td>
<td>20</td>
<td>31</td>
<td>11</td>
<td>(11/31) * 100 = 35.48%</td>
</tr>
<tr>
<td>31834</td>
<td>5</td>
<td>10</td>
<td>5</td>
<td>(5/10) * 100 = 50%</td>
</tr>
</tbody>
</table>

Figure B.1: Percent Difference between Pyannote's Predicted Speakers Compared to Ground Truth Actual Speaker Count