Thorsten Ruprechter, BSc

Increasing Efficiency In Video Transcription

Master's Thesis
to achieve the university degree of
Master of Science
Master's degree programme: Software Engineering and Management

submitted to

Graz University of Technology

Supervisor
Assoc.Prof. Dipl.-Ing. Dr.techn. Christian Gütl

Institute of Interactive Systems and Data Science
Head: Univ.-Prof. Dipl.-Ing. Dr. Stefanie Lindstaedt

Co-Supervisor
Assoc.Prof. MSc Ph.D. Foaad Khosmood
California Polytechnic State University

Graz, October 2018

Thorsten Ruprechter, BSc

Steigerung der Effizienz beim Transkribieren von Videos

Masterarbeit
zur Erlangung des akademischen Grades
Diplom-Ingenieur
Masterstudium Softwareentwicklung-Wirtschaft

eingereicht an der

Technischen Universität Graz

Betreuer
Assoc.Prof. Dipl.-Ing. Dr.techn. Christian Gütl

Institute of Interactive Systems and Data Science
Leitung: Univ.-Prof. Dipl.-Ing. Dr. Stefanie Lindstaedt

Co-Betreuer
Assoc.Prof. MSc Ph.D. Foaad Khosmood
California Polytechnic State University

Graz, Oktober 2018

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Date	Signature

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

Datum	Unterschrift

Abstract

Legislative and parliamentary proceedings present a rich source of multidimensional information that is crucial to citizens and journalists in a democratic system. At present, no fully automated solution exists that is capable of capturing all the necessary information during such proceedings. Even if professional-quality automated transcriptions existed, other tasks such as speaker identification, entity disambiguation, and rhetorical position identifications are not fully automatable. While many governments rely on expensive, manually-produced transcriptions and annotations, others are left entirely without digital transcriptions.

This thesis focuses on improving and evaluating the transcription software used by the Digital Democracy initiative, named Transcription Tool. Human transcribers work to up-level and annotate state legislative proceedings using this tool. Four phases of UI and functionality improvements are introduced and for each phase, the resulting change in efficiency is measured and presented. A logging system is developed to collect data about the transcription process. Research is performed using over 12,000 individual transcription sessions (2,300 hours of video), where each session is the record of one bill discussion. A set of about 3,200 sessions belonging to a single cohort of 20 transcribers who have experienced four versions of the Transcription Tool is further evaluated.

Through introduction of new features in the tool, human-assisted transcription efficiency can be improved by 19.4 percent over 4 phases. Furthermore, transcriber interactions with the tool are investigated. It is discovered that speaker identification, text correction, splitting and merging of utterances, tool startup, as well as transcriber inactivity are the interaction types transcription time is composed of. Lastly, more automated transformations are explored based on learning from common text corrections by transcribers. For example, an average of three name corrections can be automated per transcript using basic NLP methods. Other improvements are possible and discussed as future work.

Kurzfassung

Legislative Berichte und parlamentarische Protokolle sind Quellen von wichtigen multidimensionalen Informationen für Bevölkerung und Medien in einem demokratischen System. Zurzeit existiert keine vollautomatisierte Lösung, um alle in solchen Protokollen enthaltenen Informationen erfassen zu können. Selbst wenn es möglich wäre, hochqualitative Transkripte automatisch zu erstellen, müssten Aufgaben wie Identifikation von Rednern oder Unterscheidung von bestimmten Begriffen und Fachvokabular manuell durchgeführt werden. Viele Regierungen setzen deswegen auf teure, manuell erstellte Transkripte und Annotationen. Manche verzichten sogar völlig auf digitale Transkripte.

Diese Masterarbeit befasst sich mit der Verbesserung und Evaluierung der von der Digital Democracy Initiative verwendeten Transkribierungs-Software, genannt Transcription Tool. Menschliche Arbeitskräfte benutzen dieses Programm um Protokolle von parlamentarischen Anhörungen zu korrigieren und mit zusätzlichen Informationen zu annotieren. In dieser Arbeit werden Verbesserungen von Benutzeroberfläche und Funktionalität des Tools in vier Phasen durchgeführt. Für jede Phase wird die resultierende Effizienzänderung gemessen und präsentiert. Außerdem wird ein Logging-System eingeführt, welches Daten über den Transkribierungsprozess sammelt. Mehr als 12.000 individuelle Transkribierungssitzungen (2.300 Stunden von Videomaterial) werden untersucht, wobei in jeder Sitzung eine Diskussion über einen bestimmten Gesetzesentwurf bearbeitet wird. Eine Sammlung von ungefähr 3.200 Sitzungen, fertiggestellt von einer Gruppe von 20 Arbeitskräften, wird genauer begutachtet. Diese Gruppe hat mit allen vier entwickelten Versionen des Transcription Tools gearbeitet.

Durch Einführung der neuen Funktionalitäten kann die Effizienz des toolunterstützten, halbautomatischen Transkribierungsprozesses um 19,4 Prozent erhöht werden. Außerdem werden Benutzerinteraktionen mit dem Tool näher begutachtet. Es wird festgestellt, dass folgende Interaktionstypen existieren: Identifikation von Rednern, Textkorrektur, Aufteilen und Zusammenfügen von Aussagen, Starten des Tools sowie Inaktivität von Benutzern. Zuletzt werden Möglichkeiten für automatische Korrekturen, basierend auf häufig durchgeführten Textänderungen, untersucht. Beispielsweise können durch das Anwenden von Grundlagen der natürlichen Sprachverarbeitung durchschnittlich drei Namen pro Transkript korrigiert werden. Weitere Verbesserungsmöglichkeiten bestehen und werden konzeptionell präsentiert.

Acknowledgements

Thank you to my parents for always standing by me during my studies, whether I was in Austria or California.

I want to gratefully thank Christian Gütl for providing me with the opportunity to work on my thesis in California. Without his professional connections, this unique experience would have not been realizable for me.

I also am especially thankful to Foaad Khosmood, who cooperated with Christian Gütl to make my visit to California Polytechnic State University possible. I greatly appreciate him warmly welcoming me and always providing helpful advice, whether about this thesis or with private life in San Luis Obispo.

Furthermore, I want to thank Christine Robertson, Lisa Nunes, and Hans Poschman, staff of the Institute for Advanced Technology and Public Policy, for always helping me out when in need of administrative support.

Thank you to my fellow Transcription Tool co-workers, especially Toshihiro Kuboi and Yiupang Chan, who I intensively collaborated with during my whole stay. It was always a pleasure working with them.

In addition, thank you to all others working on Digital Democracy for giving me a home in the project lab, especially Nick Russo, Andrew Rose, Daniel Kauffman, and Michael Williams.

Without the financial support of the Austrian Marshall Plan Foundation, working on my thesis at Cal Poly would have not been feasible. I am very thankful for the support of this revered institution. An excerpt of this thesis is displayed on the Foundation's website (Marshall Plan Foundation, 2017).

A poster version of this thesis titled "Gaining Efficiency in Human Assisted Transcription and Speech Annotation in Legislative Proceedings" was accepted to the 19th Annual International Conference on Digital Government Research organized within the European commission program on Interoperability solutions for public administrations, businesses and citizens (Ruprechter, Khosmood, Kuboi, Dekhtyar, & Gütl, 2018).

This document was created using the LaTeX template provided by Voit (2018).

Contents

List of Figures

List of Tables

Listings

1. Introduction

This chapter explains the motivation behind this thesis. It also defines general objectives and provides an outline of this work.

1.1. Motivation

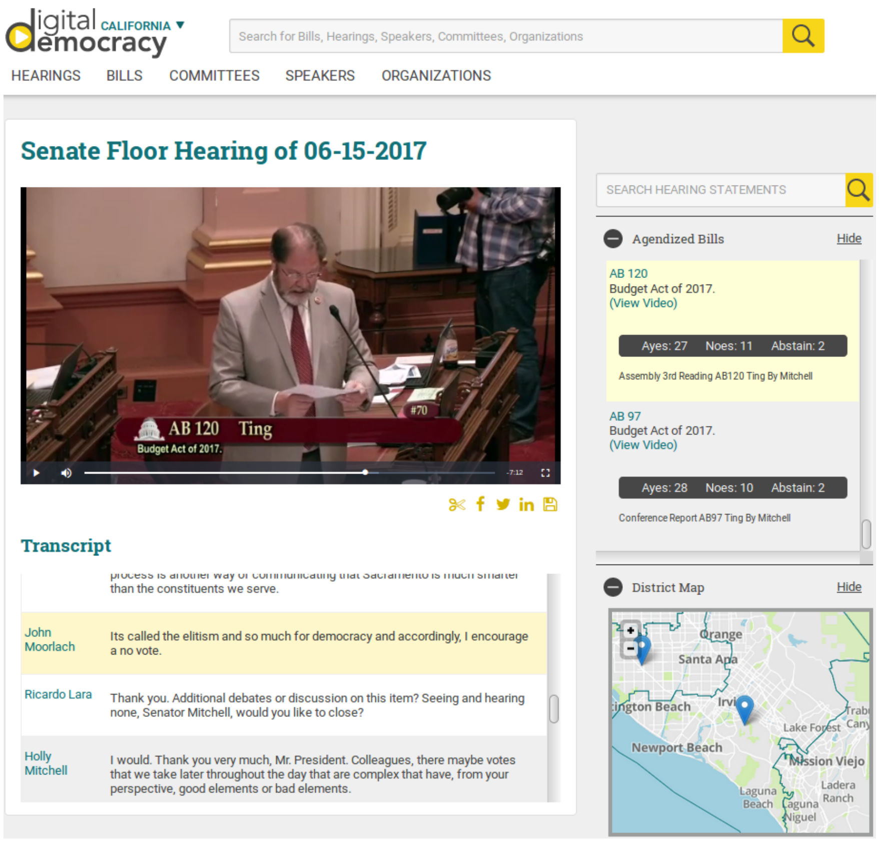
In the USA, state legislatures hold a lot of power. Unfortunately, monitoring and therefore holding them to account is difficult. It can be a time-consuming task for the general public to gain insight into the workings of state government. Official records about the content of state legislative hearings are only sparsely available.

Politically speaking, states are situated between federal and local government. State governments enforce policy and spending mandates on municipalities and local government while deciding on how budget is used. Under the US federal system, many important issues touching everyday life of citizens such as welfare, education, and law enforcement are handled by state legislatures.

Due to the massive amount of laws passed and altered each year, lobbyists are interested in influencing state legislators. Without a platform offering insight into the content of legislative hearings, the public is shut out of the process. This is especially problematic for the news media, since records about debates and bill discussions are not accessible. It is difficult to discover contents of talks and negotiations about a bill without proper textual records being created. The position of people discussing an issue might be hard to guess without knowing details about the ongoings inside state house or senate.

The Digital Democracy initiative, which was introduced in 2015, attempts to remove above-mentioned difficulties. It provides an online platform featuring professionally produced, searchable transcriptions synced with video content. This website brings transparency into state legislative hearings and government proceedings that would otherwise not be accessible to the broader public. Citizens, journalists, and researchers would not be able to easily gain information about the content of those hearings and debates.

In addition to giving insight into the proposal of new bills and laws, it also presents a chance for the general public to monitor lobbyists, lawmakers, and advocates. All information can be searched and queried, and the results are high quality thanks to human-assisted transcription. The website is accessible at www.DigitalDemocracy.org. Figure 1.1 shows how transcripts are displayed on this platform.



Senate Floor Hearing of 06-15-2017

Agendized Bills

AB 120
Budget Act of 2017.
(View Video)

Ayes: 27 Noes: 11 Abstain: 2

Assembly 3rd Reading AB120 Ting By Mitchell

AB 97
Budget Act of 2017.
(View Video)

Ayes: 28 Noes: 10 Abstain: 2

Conference Report AB97 Ting By Mitchell

Transcript

	process is another way of communicating that Sacramento is much smarter than the constituents we serve.
John Moorlach	Its called the elitism and so much for democracy and accordingly, I encourage a no vote.
Ricardo Lara	Thank you. Additional debates or discussion on this item? Seeing and hearing none, Senator Mitchell, would you like to close?
Holly Mitchell	I would. Thank you very much, Mr. President. Colleagues, there maybe votes that we take later throughout the day that are complex that have, from your perspective, good elements or bad elements.

Figure 1.1.: Interactive transcript on the Digital Democracy website.

Digital Democracy uses a human-assisted approach for generating transcription texts and metadata. While automatic transcription might be sufficient in other areas, a legislative setting requires professional and correct transcripts. This is achieved by human transcribers manually up-leveling automated transcriptions and performing annotations such as speaker and position identification. Transcribers use the Transcription Tool, a software developed by the Digital Democracy project team to enhance transcripts.

However, manual correction of transcription texts and finding speaker assignments is a tedious process. It takes human transcribers a formidable amount of time to manually go through transcripts. This bottleneck created by the manual transcription process leads to a considerable obstacle for Digital Democracy, both in regards of monetary cost as well as time delay before final transcripts are available. For this reason, improvements to the Transcription Tool are necessary.

1.2. Contributions

The main contribution of this work is the improvement of the Digital Democracy Transcription Tool as well as the introduction of metrics to measure the impact of said improvements. For this, a system is developed which collects usage information and enables performance measurements. To scientifically tackle this evaluation problem, efficiency metrics are defined.

Efficiency of the tool and newly introduced features must be analyzed and quantified. Prior to this work, there has been no reliable way for the Digital Democracy initiative to measure performance changes that new tool releases cause. However, workflow disturbances or disruptions in the transcription process introduced by new releases must be detectable. In addition, it should be possible to derive future transcription costs from efficiency results.

Another issue is the lack of clarity regarding concrete composition of transcription time. It is up to debate which aspects of transcription work yield the largest efficiency gain. To further improve performance, the crucial question of which transcriber interactions attribute how much to overall transcription time must be answered.

Lastly, potential for automation of specific aspects of the transcription process might exist. An investigation into procedures such as automatic text correction is attempted to explore these improvement possibilities.

1.3. Methodology

Content of this thesis is separated into eight chapters.

Chapter 2 provides background and related work for government transparency, transparency efforts around the world, natural language processing, transcription editing tools, and user interface evaluation.

The Digital Democracy initiative and its components are further presented in Chapter 3. An introduction is given for the official website, automated processes, the manual transcription process using the Transcription Tool, as well as past improvement projects.

Chapter 4 explains current system design and defines requirements for new implementations. For this, the main focus lies on the Digital Democracy Transcription Tool. Research problems addressed by this thesis are also elaborated.

In Chapter 5, technical implementation and tool technology changes are explained in detail. New Transcription Tool features are presented as four separate tool versions. Furthermore, implementation of a logging system is described.

Subsequently, Chapter 6 discusses experimental design and defines tool evaluation metrics. More precisely, techniques for analyzing transcription time are elaborated. This includes explanation of approaches to evaluate tool efficiency and transcriber interaction patterns. In addition, possibilities for automatic text correction are explored.

Next, Chapter 7 presents results of the analysis performed for the Transcription Tool, based on previously specified metrics.

Finally, Chapter 8 summarizes this thesis and provides conclusions as well as suggestions for future work.

2. Background and Related Work

In this chapter, literature search findings for various areas are presented. First, an introduction to government transparency and corresponding terms is given. Second, existing government transparency efforts and websites that expose legislative proceedings are listed. Next, automatic processing necessary to prepare raw data for proper presentation in such transparency services is addressed shortly. For this, focus lies on natural language processing. Then, tools used to further enrich this data are introduced. This literature survey concentrates on major tools available for transcription and annotation purposes similar to those of Digital Democracy. Lastly, some insights into common user interface evaluation techniques used to assess performance of such tools are given.

2.1. Government Transparency and Open Government

In the era of virally spreading information it is sometimes hard for the general public to distinguish between facts and the so called "fake news". Confirming claims made by politicians and actually holding them accountable can be a difficult task (Blakeslee et al., 2015). Information and data necessary to achieve this is often unstructured, for example in form of interviews or videos of political debates (Khosmood, Dekhtyar, Assai, Kurfess, & Snyder, 2014). The solution to simplify this task for citizens is to provide more transparency (Latner, Dekhtyar, Khosmood, Angelini, & Voorhees, 2017).

Government transparency is generally describable as *"publicizing of incumbent policy choices"* (Fox, 2007). Dawes and Helbig (2010) put it more precisely, when they state that it means providing possibilities to *"hold elected officials and public agencies accountable for their decisions and actions"*. In the era of the Internet, this translates to making government information easier accessible online (Khosmood et al., 2014). Discussing usage of information and communication technology (ICT) to achieve better government transparency brings up numerous recently developed terms: Open government, E-Democracy (eDemocracy), E-Government (eGovernment), and civic tech (civic technologies) are some of the most frequently occurring ones (Khosmood et al., 2014; Blakeslee et al., 2015; Latner et al., 2017; Parliamentary Office of Science and Technology, 2009; Clift, 2003; Boehner & DiSalvo, 2016). However, it is sometimes hard to distinguish between them due to varying definitions. Therefore, an attempt is made to lay out these terms in more detail below.

Open government is a phrase often used in the context of government transparency, sometimes even as a synonym (Meijer, 2015). Wijnhoven, Ehrenhard, and Kuhn (2015) define it as two dimensions. On the one hand, it should provide access to information. On the other hand, it should also enable citizens to participate in decision making. These authors also mention that the idea of open government is often simplified to collaboration of the public sector with the crowd. According to Janssen, Charalabidis, and Zuiderwijk (2012), open government should interact with its environment and act as an open system. Furthermore, the authors agree with Wijnhoven et al. (2015) that it must allow effective oversight by promoting transparency and participation. Khosmood et al. (2014) as well as Dawes and Helbig (2010) also mention the characteristic goals of the open government principles outlined by the US government under President Obama: collaboration, participation, and transparency.

E-Democracy usually describes increasing and enhancing the public's engagement in politics by using ICT (Parliamentary Office of Science and Technology, 2009; Clift, 2003). Clift (2003) also adds that E-Democracy involves utilizing specific strategies by democratic actors. According to the authors, this includes political parties and interest groups as well as the government itself. In addition, independent actors like civil society organizations or other initiatives organized by citizens and voters may also embrace E-Democracy.

The term E-Government is mostly used when talking about better delivery of government services to citizens utilizing ICT (Parliamentary Office of Science and Technology, 2009). Those services should be personalized to citizen's needs, not those of their provider (Parliamentary Office of Science and Technology, 2009). Furthermore, some authors describe E-Government broadly as *"E-Democracy activities of government institutions"* (Clift, 2003). It is arguable that in the context of term descriptions given here, E-Government can be seen as a sub-topic of E-Democracy.

Civic tech (or civic technologies) is interpretable as the technology and platforms opening up government (Knight Foundation, 2015). One could say they enable E-Democracy. According to the Knight Foundation (2015), further goals for civic tech are: building place-based social capital, increasing civic engagement, promoting deliberative democracy, as well as fostering inclusion and diversity. While government organizations also develop such technologies, they are often initiated by citizen-lead initiatives to provide transparency services immune to manipulation. Boehner and DiSalvo (2016) loosely define civic tech as the *"design and use of technology to support both formal and informal aspects of government and public services"*.

Table 2.1 summarizes the terms explained in the previous paragraphs and lists sources for their definitions.

Terms	**Definitions**	**Sources**
Government Transparency	Publicizing of incumbent policy choices; Providing possibilities to hold elected officials and public agencies accountable for their decisions and actions	Fox (2007) Dawes and Helbig (2010)
Open Government	Providing access to information and enabling citizens to participate in decision making; Synonym for Government Transparency; Government enabling citizen participation, collaboration, and transparency	Wijnhoven, Ehrenhard, and Kuhn (2015) Janssen, Charalabidis, and Zuiderwijk (2012) Meijer (2015) Khosmood, Dekhtyar, Assai, Kurfess, and Snyder (2014) Dawes and Helbig (2010)
E-Democracy (eDemocracy)	Increasing and enhancing the public's engagement in politics by using ICT	Parliamentary Office of Science and Technology (2009) Clift (2003)
E-Government (eGovernment)	Delivery of government services utilizing ICT; E-Democracy activities of the government	Parliamentary Office of Science and Technology (2009) Clift (2003)
Civic Tech (Civic Technologies)	Technology and platforms opening up government; Design and use of technology to support both formal and informal aspects of government and public services	Knight Foundation (2015) Boehner and DiSalvo (2016)

Table 2.1.: Terms commonly used in context of government transparency.

2.2. State of Government Transparency Around the World

Government transparency policies and measures vary in different parts of the world. Below some of the transcription systems which allow monitoring of lawmakers in the USA, the Commonwealth of Nations, and Europe are investigated.

The situation regarding government transparency in USA state governments is problematic, as already mentioned in Chapter 1. Exact implementations of measures ensuring transparency differ from state to state. For this thesis, focus is laid on explorations regarding the state of California. Rovin (2016) mentions the F grade given to California by Davis and Baxandall (2014) in a state comparison of access to online information regarding spending information. In this report published by the CALPIRG Education Fund, California placed last. Since then, another investigation mandated by the CALPIRG Education Fund was carried out by Surka and Ridlington (2016). While other states improved their transparency policies and websites, California again came in last out off all states and received an F grade. This further showcased the still existing need for a publicly accessible platform providing in-depth information about political proceedings.

Until 2018, no improvements to transparency policies were attempted by the state. Although a recent proposition guarantees availability of videos and closed captions, these records are severely lacking in quality (Legislative Analyst's Office, 2016). Some publicly available resources to get insight into hearings exist, but most of them do not offer rich functionalities like full-text search in transcripts (Rovin, 2016; Khosmood et al., 2014; Blakeslee et al., 2015). For example, The California Channel (2017) hosts a video on demand archive for hearing videos. However, this service is not maintained by the state government and receives no state funding. The archive and corresponding live channel are run by California's cable television operators. Furthermore, the State of California (2017c) offers a live stream as well as a media archive which is searchable in a similar fashion to the California Channel archive. Detailed information about senate committees is also available. However, the State of California (2017c) does not create textual transcripts of hearings. In addition, the State of California (2017a) offers the same information about the California State Assembly sessions and committees on a separate website. Another official source of information for citizens is the California State Legislature Website (State of California, 2017b). It holds information about bills and allows for full-text search of bill texts. Various information about lobbying can be retrieved from the websites of the California Secretary of State (2017) and MapLight (2017). Finally, Open States (2017) provides access to legislator district data.

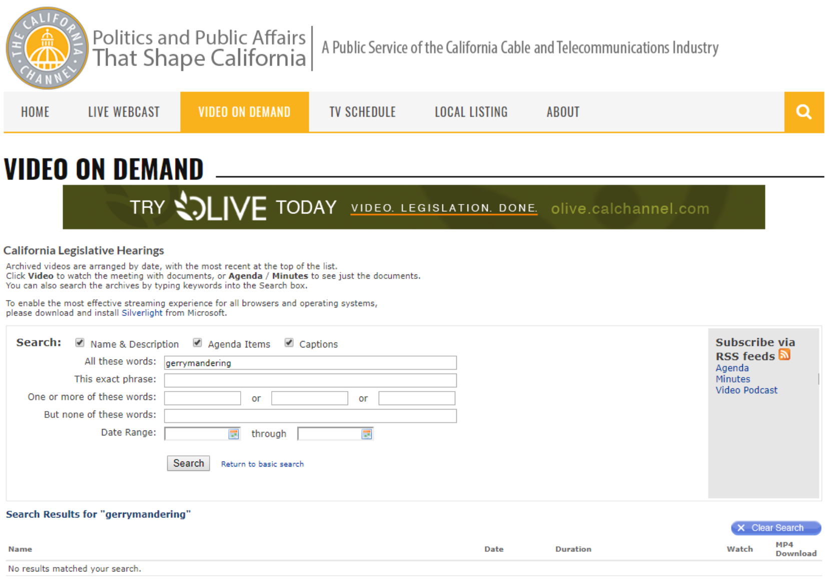
To solve the problem of missing disclosure in state governments, the Digital Democracy initiative was instigated (Khosmood et al., 2014). The main goal of this initiative is to provide full insight and access into US state legislative processes. Videos of legislative committee hearings are combined together with auxiliary information such as searchable transcriptions, bills discussed in hearings, and identification of participating speakers like legislators, lobbyists, witnesses, and members of the general public. For legislators, full legislative service history as well as campaign contributions and gift data are tabulated and presented to the user. The Digital Democracy initiative created and maintains technologies which link all of this information together (Khosmood et al., 2014). Practical usage of the website presenting the information can be examined by visiting www.DigitalDemocracy.org. As of this writing, Digital Democracy processes information for four states: California, Florida, New York, and Texas. Figure 2.1 displays an comparison of the Digital Democracy search functionality with the video on demand archive of The California Channel (2017). For more information on Digital Democracy, see Section 3.

Although American state governments do not usually provide searchable transcripts of hearings to the general public, resources are available for congressional proceedings of the federal government. C-SPAN is a cable TV channel that broadcasts congressional hearings and creates searchable transcripts which are linked to the video source (C-SPAN, 2018). Hearing records include vote, bill, and speaker data. This archive provided by C-SPAN allows for search in federal proceedings of Congress, the Executive Branch, as well as the Supreme Court. In addition, citizens can access proceedings and debates online through the Congressional Record website (Library of Congress, 2018).

The British parliament provides official transcripts of parliamentary debates in a searchable manner. These records, officially called "Hansard", are accessible to the public via the parliament's online presence (Parliament of the United Kingdom, 2018). During a sitting day, an online version of this day's proceedings is published gradually, with the full Hansard being available the next morning. Similar to Digital Democracy, the website provides a search interface which allows to query transcripts for specific terms. It directly links the search results to the exact position in both textual transcripts and video recordings. Figure 2.2 shows an example of the comprehensible search interface provided by the British government. Some of the Commonwealth countries also implement searchable Hansards. For example, the Parliament of Australia (2018), the New Zealand Parliament (2018), and the Parliament of Canada (2018) all offer Hansards on their websites while creating links between plain text records and corresponding videos. They also integrate advanced interfaces for multi-purpose search.

The European Union provides online access to debate videos and verbatim texts (The European Parliament, 2018). However, segments of speech in textual transcripts of debates are not translated to all languages. This effectively means that debate transcripts only contain speaker utterances in the original spoken language, which is mostly not English. Although speeches contained in reports are linked to video sources and therefore translated audio, no full text search of plenary sittings is available. On the other hand, agendas, reports, adopted texts, votes, and audio recordings of debates are made available in all official languages of the EU.

Similar to the EU, the parliament of Switzerland only creates textual transcripts in the original language of the speaker. However, a searchable official bulletin is made available which allows users to issue full-text queries on statements of council


Politics and Public Affairs That Shape California
A Public Service of the California Cable and Telecommunications Industry
HOME
LIVE WEBCAST
VIDEO ON DEMAND
TV SCHEDULE
LOCAL LISTING
ABOUT
TRY LIVE TODAY VIDEO. LEGISLATION. DONE. olive.calchannel.com
Search
Return to basic search
Clear Search

VIDEO ON DEMAND

California Legislative Hearings

Archived videos are arranged by date, with the most recent at the top of the list.
Click **Video** to watch the meeting with documents, or **Agenda** / **Minutes** to see just the documents.
You can also search the archives by typing keywords into the Search box.

To enable the most effective streaming experience for all browsers and operating systems, please download and install Silverlight from Microsoft.

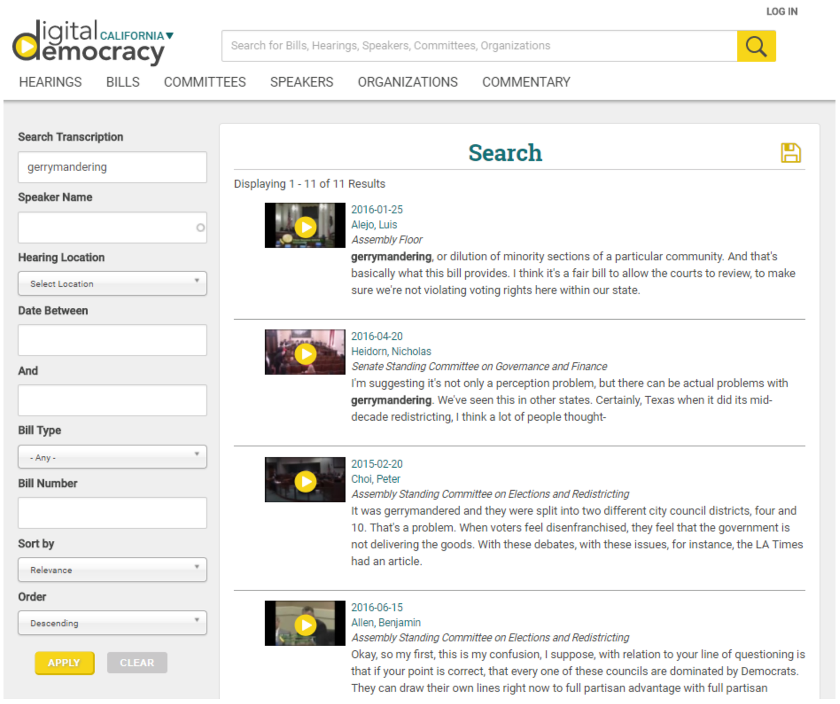
Search: [✓ Name & Description] [✓ Agenda Items] [✓ Captions]
All these words: gerrymandering
This exact phrase:
One or more of these words: or or
But none of these words:
Date Range: through

Subscribe via RSS feeds
Agenda
Minutes
Video Podcast

Search Results for "gerrymandering"

Name	Date	Duration	Watch	MP4 Download
No results matched your search.				

(a) Search Form of the California Channel Video On Demand Service


LOG IN
digital CALIFORNIA democracy
Search for Bills, Hearings, Speakers, Committees, Organizations
HEARINGS
BILLS
COMMITTEES
SPEAKERS
ORGANIZATIONS
COMMENTARY
APPLY
CLEAR

Search Transcription: gerrymandering
Speaker Name:
Hearing Location: Select Location
Date Between:
And:
Bill Type: - Any -
Bill Number:
Sort by: Relevance
Order: Descending

Search

Displaying 1 - 11 of 11 Results

2016-01-25
Alejo, Luis
Assembly Floor
gerrymandering, or dilution of minority sections of a particular community. And that's basically what this bill provides. I think it's a fair bill to allow the courts to review, to make sure we're not violating voting rights here within our state.

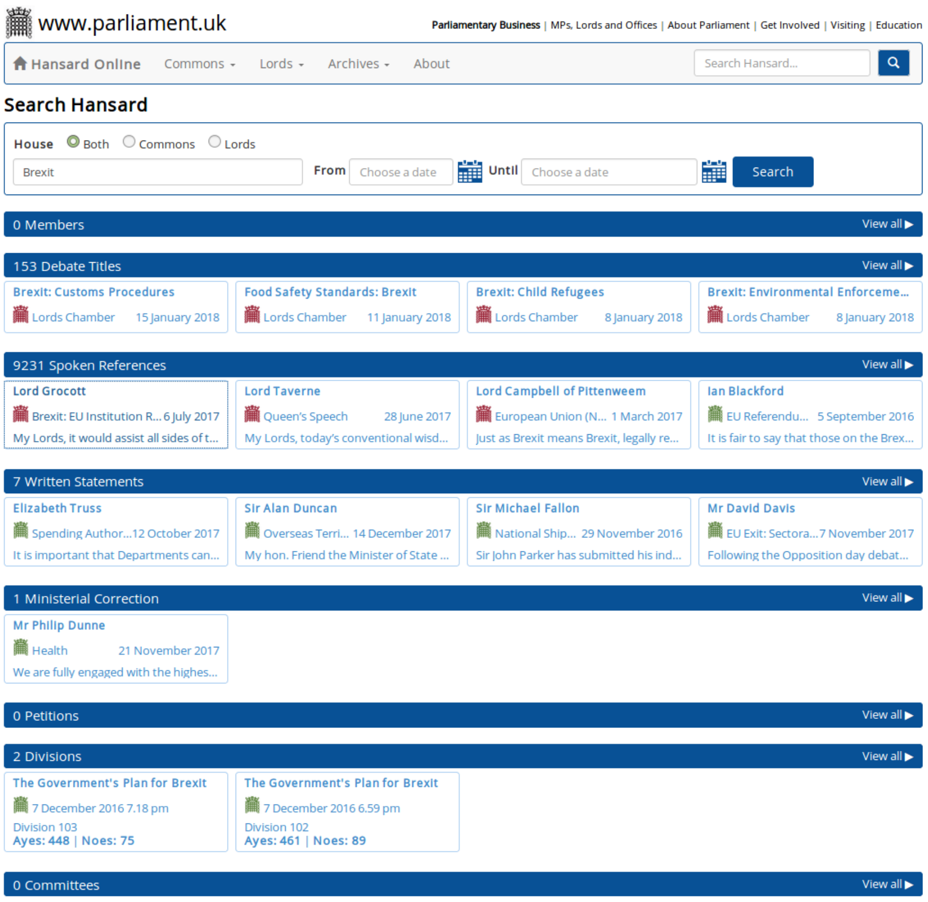
2016-04-20
Heidorn, Nicholas
Senate Standing Committee on Governance and Finance
I'm suggesting it's not only a perception problem, but there can be actual problems with **gerrymandering**. We've seen this in other states. Certainly, Texas when it did its mid-decade redistricting, I think a lot of people thought-

2015-02-20
Choi, Peter
Assembly Standing Committee on Elections and Redistricting
It was gerrymandered and they were split into two different city council districts, four and 10. That's a problem. When voters feel disenfranchised, they feel that the government is not delivering the goods. With these debates, with these issues, for instance, the LA Times had an article.

2016-06-15
Allen, Benjamin
Assembly Standing Committee on Elections and Redistricting
Okay, so my first, this is my confusion, I suppose, with relation to your line of questioning is that if your point is correct, that every one of these councils are dominated by Democrats. They can draw their own lines right now to full partisan advantage with full partisan

(b) Digital Democracy Search Functionality

Figure 2.1.: Comparison between the search functionality of the California Channel video on demand service and the Digital Democracy website using the keyword "gerrymandering".



www.parliament.uk

Parliamentary Business | MPs, Lords and Offices | About Parliament | Get Involved | Visiting | Education

Hansard Online · Commons · Lords · Archives · About · Search Hansard...

Search Hansard

House: [✓ Both] Commons Lords

Brexit · From: Choose a date · Until: Choose a date · Search

0 Members — View all

153 Debate Titles — View all

Brexit: Customs Procedures	Food Safety Standards: Brexit	Brexit: Child Refugees	Brexit: Environmental Enforceme...
Lords Chamber 15 January 2018	Lords Chamber 11 January 2018	Lords Chamber 8 January 2018	Lords Chamber 8 January 2018

9231 Spoken References — View all

Lord Grocott	Lord Taverne	Lord Campbell of Pittenweem	Ian Blackford
Brexit: EU Institution R... 6 July 2017	Queen's Speech 28 June 2017	European Union (N... 1 March 2017	EU Referendu... 5 September 2016
My Lords, it would assist all sides of t...	My Lords, today's conventional wisd...	Just as Brexit means Brexit, legally re...	It is fair to say that those on the Brex...

7 Written Statements — View all

Elizabeth Truss	Sir Alan Duncan	Sir Michael Fallon	Mr David Davis
Spending Author...12 October 2017	Overseas Terri... 14 December 2017	National Ship... 29 November 2016	EU Exit: Sectora...7 November 2017
It is important that Departments can...	My hon. Friend the Minister of State ...	Sir John Parker has submitted his ind...	Following the Opposition day debat...

1 Ministerial Correction — View all

Mr Philip Dunne
Health 21 November 2017
We are fully engaged with the highes...

0 Petitions — View all

2 Divisions — View all

The Government's Plan for Brexit	The Government's Plan for Brexit
7 December 2016 7.18 pm	7 December 2016 6.59 pm
Division 103	Division 102
Ayes: 448 | Noes: 75	Ayes: 461 | Noes: 89

0 Committees — View all

Figure 2.2.: Screenshot of an example search using the interactive Hansard on the British Parliament website.

members (in their original language) and provides links to transcripts and videos (The Swiss Parliament, 2018).

In Austria, stenographic protocols of plenary sittings of the parliament are made available online (The Austrian Parliament, 2018b). The full-text of these records can be searched using the advanced search functionality provided on the website of the Austrian parliament, but there is no direct link from found text to video recordings (The Austrian Parliament, 2018a). Currently, there also exists no permanent video archive of past parliamentary sittings. However, sittings are streamed live on the parliamentary website and are stored for seven day on-demand access in the archive of the Austrian national public service broadcaster (ORF, 2018). Most of the functionality and information is only available in German. Since the official services of the parliament in Austria do not keep track of votes, the website Addendum.org provides this service to the public by manually counting the physical votes of the representatives attending a sitting (Quo Vadis Veritas Redaktions GmbH, 2018).

Table 2.2 shows an overview and summary of how the aforementioned governments and parliamentary systems provide information to the public. This overview makes it obvious that California as a state government lacks official records. Surprisingly, Austria also stands out due to the lack of an extensive media archive and official vote data. Lastly, it has to be stated that user interface for many parliamentary websites is sub-par. In the case of Switzerland and Austria, the search filter can only be reached by navigating through various menus.

Government	**Records**				**Search Filter/Scope**			
	Transcript	**Videos**	**Linked**	**Votes**	**Full-Text**	**Title**	**Date**	**Speaker**
European Union	OL	✓	✓	✓		✓	✓	✓
Australia	✓	✓		✓	✓	✓	✓	✓
New Zealand	✓	✓	✓	✓		✓	✓	✓
Canada	✓	✓	✓	✓	✓	✓	✓	✓
United Kingdom	✓	✓	✓	✓	✓	✓	✓	✓
Switzerland	OL	✓		✓	✓	✓	✓	✓
Federal Government of the USA	✓	✓		✓	✓	✓	✓	✓
State of California		✓						
DigitalDemocracy (CA, FL, NY, TX)	✓	✓	✓	✓	✓	✓	✓	✓
Austria	✓	< 7 days		TP	✓	✓	✓	

Table 2.2.: Summarization of the survey conducted in Section 2.2 about government transparency around the world (OL = only available in original spoken language, TP = third-party sites).

2.3. Natural Language Processing

Section 2.2 introduced government transparency systems providing information about legislative proceedings. However, data has to be preprocessed before it can be properly presented on such platforms. More often than not, systems combine manual and automated processing steps to enrich proceedings (Khosmood et al., 2014). This is important, since records of political hearings have to be as immaculate as possible. Automated processes utilize technology such as speech-to-text algorithms, face recognition and image processing, as well as natural language processing (NLP) (Kauffman, Williams, Washington, Socher, & Khosmood, 2018; Budhwar, Kuboi, Dekhtyar, & Khosmood, 2018; Blakeslee et al., 2015). Due to their relevance for this thesis, specific concepts of NLP are investigated further in the following subsections.

2.3.1. Word Similarity and Distance Metrics

Many approaches to compare strings exist. Taken from studies by Christen (2006), Cohen, Ravikumar, and Fienberg (2003), and Snae (2007), a selection of some of the most commonly used algorithms is collected:

- Levenshtein
- Damerau–Levenshtein
- Jaro–Winkler
- Bag distance
- Guth
- Longest common sub-string (LCS)
- Monge–Elkan
- Jaccard
- TF-IDF
- Soundex
- NYSIIS

Explaining all of these would go beyond the scope of this paper. However, a closer look at the frequently used Levenshtein and Jaro-Winkler algorithms is taken in the following sections.

Levenshtein

First introduced by Levenshtein (1966), this distance metric has become a popular way of measuring differences in words. Superficially, it describes the number of deletions, insertions, and substitutions of characters needed to convert one word into another. Each of these operations usually bears a cost of 1. However, some implementations allow definition of a specific substitution cost, such as in Python's Natural Language Toolkit (NLTK) (NLTK Project, 2017b).

A simple distance value like this might not be significant enough on its own. This is especially relevant for Levenshtein, since word length is not taken into consideration at all. Therefore, instead of only utilizing distance, a ratio can also be calculated. The formula for this Levenshtein ratio can be seen in Equation 2.1. This ratio was modeled after the definitions for normalized edit distances by Marzal and Vidal (1993). It is computed by dividing the Levenshtein distance of two words by length of the longer word. The formula basically represents the percentage of the longer word which has to be changed in order to transform it into the shorter one, considering the Levenshtein distance.

$$L_{ratio_{w_1,w_2}} = \frac{L_{distance}(w_1, w_2)}{max(len(w_1), len(w_2))} \tag{2.1}$$

Jaro–Winkler

Another way of comparing strings is the Jaro–Winkler distance (or similarity). It was first presented by Winkler (1990), based on the work of Jaro (1989).

Jaro (1989) proposed a similarity value which considers matching characters in two strings and the transpositions necessary to transform one word into another. Winkler (1990) extended the algorithm so that differences in the beginnings of strings (prefixes) significantly influence the result. This approach is based on the idea that words starting with the same few characters tend to be more similar than those which do not. Yancey (2005) summarizes formulas and provides examples for calculating string distance metrics.

2.3.2. Sequence Alignment

Some similarity algorithms, such as those mentioned in Section 2.3.1, are mostly utilized to detect changes while not considering alignment in a given context. However, in some cases the global and local alignment of sequences might be relevant. Rovin used the Needleman–Wunsch algorithm as a metric for similarity in transcription texts (see Section 3.5.2). This algorithm considers alignment of texts when calculating differences. It is closer elaborated in the section below.

Needleman–Wunsch Algorithm

General This dynamic programming algorithm was first introduced by Needleman and Wunsch (1970). Back then, its main purpose was to compare proteins by their amino acid sequences and find their best possible alignment. This is achieved by attempting to maximize similarity between sequences. Since proteins are represented by a number of amino acids which are ordered, they can be depicted as textual strings. In these strings, each amino acid is represented by one letter. In addition to its use in bioinformatics, Needleman–Wunsch is well-suited for comparing words or sentences in NLP.

To compare two sequences (P and Q) represented by strings of letters, a two-dimensional-array (A) is build. The array has a width of $length(p) + 2$ and height of $length(q) + 2$. In the first row and first column the two sequences are written down, respectively starting in the third cell. To initialize the array, cell $A_{1,1}$ is filled with a zero. Then, cells both below and right are successively decreased by one.

For each cell, three possible cases exist. Needleman and Wunsch defined a scoring system for these cases. First, the two currently compared letters match (+1). Second, the letters do not match (-1). Third, a letter is deleted or inserted (-1) – this defines a gap, sometimes also called spacing. However, the scoring system can be modified and extended. An example for such an alternative scoring system is given in Section 2.3.2 below.

When filling a cell with a new value, one has to consider contents of adjacent cells. In case of a character mismatch, the mismatch score is added to the value in the top-left cell, while values to the top and left have to be increased by the gap score (spacing) score. If two letters match, the same scoring policy can be applied to all adjacent cells. In either case, the value of the currently calculated cell is then chosen to be the lowest out of the three calculated sums. This procedure is continued from the array's top-left to the bottom-right until all cells are filled. It is important to keep track of which cell was chosen as the source of each new value. This is necessary for identifying the best possible alignment which maximizes similarity.

Usage by Rovin Rovin (2016) used the Needleman–Wunsch algorithm to compare transcription texts during evaluation of the improvements introduced by him (see Section 3.5.2). However, he introduced some changes to the standard approach described in Section 2.3.2 above. Instead of comparing single letters, the modified algorithm compares sentences in transcripts. In his version, the first row and column is made up of full transcription texts. Each word occupies one cell.

In consideration of the usage of full words instead of letters, Rovin adapted a scoring system with the following four cases and respective scores:

1. **Full match (0):** Both words match completely.
2. **Partial match (1):** Words match after removal of non-alphanumeric characters and conversion to lower-case.
3. **Mismatch (5):** Words do not match.
4. **Space/Gap Insertion (3):** Shifting words would result into a cheaper transcription.

In the end, the cheapest path is chosen, thus leading to the optimal alignment between transcription texts. From this alignment, percentage of matched words is calculated and chosen as an evaluation metric.

Other Algorithms

Besides the Needleman–Wunsch algorithm described in Section 2.3.2, other sequence alignment algorithms exist.

For example, the Smith–Waterman algorithm is a variation of the Needleman–Wunsch algorithm which strongly focuses on local alignment (Smith & Waterman, 1981).

Hirschberg (1975) implemented a more efficient algorithm for sequence alignment which utilizes divide and conquer as well as the distance function developed by Levenshtein (1966).

2.3.3. Part-Of-Speech Tagging

According to Oxford University Press (2017), part-of-speech (POS) is *"a category to which a word is assigned in accordance with its syntactic functions"*. Corresponding categories existing in English grammar are noun, pronoun, adjective, determiner, verb, adverb, preposition, conjunction, and interjection. In linguistics, part-of-speech may also be referred to as word class. In computer science, these categories are often called POS-tags.

Subsequently, the process of automatically applying tags to words in a text is called POS-tagging. Taggers can be implemented in various ways. Examples for early solutions to this problem are rule-based, transformation-based, or memory-based tagging methods (Brill, 1995; Daelemans, Zavrel, Berck, & Gillis, 1996). In addition, taggers realizing statistical methods and machine learning, for example Hidden Markov Models, maximum-entropy approaches, circular dependency networks, or conditional random fields exist (Brants, 2000; Cutting, Kupiec, Pedersen, & Sibun, 1992; Berger, Pietra, & Pietra, 1996; Toutanova & Manning, 2000; Toutanova, Klein, Manning, & Singer, 2003; Lafferty, McCallum, & Pereira, 2001).

Practically speaking, some of the easiest accessible POS-taggers are available through Python's NLTK (NLTK Project, 2017a), as well as the Stanford POS-tagger (Stanford Natural Language Processing Group, 2018a) which is based on the work of Toutanova et al. (2003). These taggers are commonly used for academic research. An alternative to these implementations is provided by the industrial-level framework spaCy (Explosion AI, 2018). A resource worth mentioning is SyntaxNet (Google LLC, 2018). This toolkit for Google's TensorFlow framework (Martin Abadi et al., 2015) also offers a parser which is able to perform POS-tagging. Another framework provided by a major software organization which includes a POS-tagger is Apache OpenNLP by The Apache Software Foundation (2018a).

2.3.4. Named-Entity Recognition

According to Tjong Kim Sang and De Meulder (2003), named entities are *"phrases that contain the names of persons, organizations and locations"*. Nadeau and Sekine (2007) describe named-entity recognition (NER) as a task to *"recognize information units like names, including person, organization and location names, and numeric expressions including time, date, money and percent expressions"*.

Handcrafting rules for specific languages, areas of applications, or domains and their corresponding entities is a tedious task. Because of this, most contemporary NER systems are based on statistics and machine learning (Nadeau & Sekine, 2007). However, even in recent years new rule-based systems for some domains emerged (Eftimov, Koroušić Seljak, & Korošec, 2017; Alfred, Leong, On, & Anthony, 2014). When talking about NER utilizing machine learning, one can categorize methods into three topics: supervised, semi-supervised, and unsupervised (Nadeau & Sekine, 2007).

First, as in classical machine-learning, supervised systems use large annotated text corpora to learn how entities are recognized. Next, semi-supervised learning is a commonly mentioned procedure in NER. For this approach, a small degree of supervision is needed to start off the learning process. Initially, a few either automatically mined or manually defined seed entities are used in the early phase of the learning process. Automatically mined data could be pulled from websites, dictionaries, or other sources. Then, words and phrases occurring in similar contexts as these well-defined entities are assigned to the same information unit (Nadeau & Sekine, 2007; McCallum & Li, 2003). This process is sometimes also referred to as "bootstrapping". In unsupervised systems, algorithms attempt clustering of entities which occur in a similar context. Then, labels for these clusters are found. This is achieved by either accessing lexical resources such as dictionaries or WordNet, web searches, or by their context itself. For example, if an entity is commonly followed by "is a city" or "was born in", then its entity group can be derived from that information (Nadeau & Sekine, 2007; Ratinov & Roth, 2009).

When looking at implementations of NER systems, similar frameworks to those listed for POS-tagging above can be mentioned (see Section 2.3.3). NLTK, the Stanford NER-tagger, spaCy, and Apache OpenNLP all offer capabilities to perform NER (NLTK Project, 2017a; Stanford Natural Language Processing Group, 2018b; Explosion AI, 2018; The Apache Software Foundation, 2018a).

Table 2.3 exemplifies POS- and NER-tagging of a sentence. Of course, more tags than those used in the table exist. However, this example is purposefully kept simple. Different taggers also tag data in various ways. For example, in the corresponding sentence some taggers might mark "Thorsten" as an organization, dependent on the language model. Similarly, "Cal Poly" could also be tagged as a location when using a different model.

Table 2.3.: Example sentence with POS and NER tags.

Word	Thorsten	visited	Cal	Poly	in	2018	.
POS-tag	NNP	VBD	NNP	NNP	IN	CD	.
POS Description	Proper Noun	Verb	Proper Noun	Proper Noun	Adposition	Cardinal Number	Punctuation
NER-Tag	PER	O	ORG	ORG	O	DATE	O
NER Description	Person	Not-An-Entity	Organization (Begin)	Organization (End)	Not-An-Entity	Date	Not-An-Entity

2.4. Transcription Editing and Annotation Tools

Although automatic algorithms nowadays manage to generate reasonable transcripts as well as additional contextual data from digital sources, human-assistance is still mostly necessary to ensure correctness on a professional level (Blakeslee et al., 2015). Due to this, tools for both research and commercial use emerged in the past years. However, transcription software differs strongly from field to field.

Entity tagging or annotation tools such as introduced by Stenetorp et al. (2012), Papazian, Bossy, and Nédellec (2012), or Widlöcher and Mathet (2012) focus on linking metadata to plain text. They allow users to create links between existing or newly created entities as well as their text occurrences. In addition, further details such as description of relationships can be specified. Although such annotation tools fulfill a different purpose than transcription tools do overall, both generate metadata for plain text records. Entity information created by transcription tools can be used to derive more in-depth information about interactions and relationships existing in the current setting. Such information could for example be represented by speaker identification and speaker alignment regarding a currently discussed issue.

Audio transcription tools must allow the user to pause, rewind, or in any other way manipulate the currently investigated files while editing transcripts. Even though audio tracks are a different media type than video, many audio transcription systems are similar to those handling video in that the initial text presented to the user is also created by automatic speech recognition (Luz, Masoodian, Rogers, & Deering, 2008; Burke, Amento, & Isenhour, 2006; Revuelta-Martinez, Rodriguez, & Garcia-Varea, 2012; Whittaker & Amento, 2004; Basu, Bepari, Nandi, Khan, & Roy, 2013). Some tools provide the option to investigate waveform representation of played audio, such as the one implemented by Luz et al. (2008). The main difference between transcription tools developed for audio and video is that the latter could be used to derive additional information. An example for such information is that it might be easier to identify speakers based on their physical on-screen appearance than by voice or text alone.

Software which assists users in transcription of handwriting or ancient texts has to properly handle display and navigation of texts or still images, in order that single words and characters can be properly deciphered (Toselli, Vidal, & Casacuberta, 2011; Castro-Bleda et al., 2017). These tools use raw transcripts produced by optical character recognition technologies as their primary source of text. It is also planned to incorporate image analysis in the preprocessing pipeline of the Digital Democracy Transcription Tool, where results of automatic face recognition could help transcribers to identify speakers (see Section 3.3).

The subset of tools most similar to the Digital Democracy Transcription Tool (see Section 3.4) are those used to create and edit video subtitles, captions, or transcripts. Although subtitle and caption editors do not belong to the exact same category as those focusing on transcription texts, they still provide valid input for user interface decisions and features implemented in a transcription tool. One of the earliest approaches of providing a software package for creating and synchronizing video transcripts was introduced by Nivre et al. (1998). The authors chose to split up the main components into separate tools. *TransTool* is used to generate the transcription texts while *SyncTool* synchronizes transcripts with video recordings by enabling the user to manually set time codes. Seps (2013) created *NanoTrans*, a tool which allows for creation of both textual and phonetic transcriptions. Besides the usual approach of providing UI panels for both transcription and video, *NanoTrans* also includes a panel visualizing the audio track in waveform. In addition, a button panel is available to insert non-speech event tags into the transcript. A different approach was taken by Deshpande, Tuna, Subhlok, and Barker (2014), who developed *ICS Caption Editor*, a crowd-sourced caption generator which enables students to collaborate on correction of preprocessed captions generated from lecture videos. To improve quality of texts, users can request a review on problematic or complicated sections directly through the editing interface.

Besides these research-oriented transcription tools, commercial software exists which provides assistance in editing video transcripts, captions, or subtitles. While most of these tools share the same functionality, some have unique features or user interfaces. NowTranscribe Ltd (2017) offers an audio transcription tool which shows the automatically generated transcripts in light gray. Pressing the tab key accepts the currently displayed word while pressing any other key allows the user to modify the text. The automatic transcription services provided by cielo24 (2018) come with a sophisticated transcription editor. This tool is the one most similar to the Digital Democracy Transcription Tool functionality-wise. Some of the features of this editor are: allowing users to navigate between utterances using a button panel, jumping to specific parts of a video by entering a timestamp, adding speaker information to utterances, the option of auto-pausing the video player while a user is typing, hotkeys for navigating the transcript, and shortcuts for adding sound tags.

To summarize the findings above, one can state that the following categories of transcription editing and annotation tools exist:

- Entity tagging and linking metadata to text
- Handwriting and ancient text transcription
- Audio transcription
- Subtitle editing
- Caption editing
- Video transcription

However, it may be hard to clearly distinguish between categories in some cases. For example, software used to transcribe audio may also be useful to create and export subtitles or video captions. Dependent on exact tool features and requirements, the same could be said about subtitle, caption, and video transcription editors. Example use cases for such editors include: records of lectures or educational videos (Deshpande et al., 2014; Seps, 2013), subtitles and captions for television, movies, or presentations (cielo24, 2018), rich transcriptions of legislative hearings (Rovin, 2016), or any other arbitrary video resource.

2.5. User Interface Evaluation

Developing and evaluating tools such as those introduced in Section 2.4 requires well-defined processes. For this, user interface (UI) testing and usability evaluation methods have to be applied. Therefore, this section explores some of these techniques. While an overview of specific UI evaluation methods is given first, a broader look on the topic is taken afterward to gain better understanding of general concepts.

2.5.1. Overview of User Interface Evaluation Methods

In the past few decades, many approaches for user interface evaluation emerged. Ivory and Hearst (2001) present a survey which addresses various methods, some of them being:

- Thinking-Aloud Protocol
- Log File Analysis
- Cognitive Walkthrough
- Heuristic Evaluation
- Interviews, Surveys, Questionnaires
- Programmable User Models

This list only shows a small number of techniques. Table 2.4, an excerpt of Table 2 in Ivory and Hearst (2001), specifies an extensive list of such methods with a brief explanation of how they are usually conducted.

As it is visible from this list, a vast amount of different approaches exists. It would be an excessive task to investigate all of them right away. Due to this reason, a step back is taken first. A broader classification according to data sources (data collection) is attempted in Section 2.5.2 below to gain a better idea of this topic first.

2.5.2. Categorization of Evaluation Methods by Data Sources

There are multiple ways of categorizing UI evaluation methods. Some authors, like Dumais, Jeffries, Russell, Tang, and Teevan (2014), differentiate methods for data gathering for UI testing into lab studies, field studies, and log studies.

During lab studies, participants are asked to perform specifically designed tasks in a well-defined set-up. Dumais et al. (2014) argue that while such studies are the most controlled approach, they might not capture real-world user experience. This mainly stems from the artificial environment lab studies provide. In contrast to that, lab studies make it easy to focus on interesting questions researchers want to

Table 2.4.: User interface evaluation methods and their short description. Adapted from Table 2 by Ivory and Hearst (2001).

Method Name	**Short Description**
Thinking-Aloud Protocol	User talks during test
Question-Asking Protocol	Tester asks user questions
Shadowing Method	Expert explains user actions to tester
Coaching Method	User can ask an expert questions
Teaching Method	Expert user teaches novice user
Co-discovery Learning	Two users collaborate
Performance Measurement	Tester records usage data during test
Log File Analysis	Tester analyzes usage data
Retrospective Testing	Tester reviews videotape with user
Remote Testing	Tester and user are not co-located during test
Guideline Review	Expert checks guideline conformance
Cognitive Walkthrough	Expert simulates user's problem solving
Pluralistic Walkthrough	Multiple people conduct cognitive walkthrough
Heuristic Evaluation	Expert identifies violations of heuristics
Perspective-Based Inspection	Expert conducts narrowly focused heuristic evaluation
Feature Inspection	Expert evaluates product features
Formal Usability Inspection	Expert conducts formal heuristic evaluation
Consistency Inspection	Expert checks consistency across products
Standards Inspection	Expert checks for standards compliance
Contextual Inquiry	Interviewer questions users in their environment
Field Observation	Interviewer observes system use in users environment
Focus Groups	Multiple users participate in a discussion session
Interviews	One user participates in a discussion session
Surveys	Interviewer asks user specific questions
Questionnaires	User provides answers to specific questions
Self-Reporting Logs	User records ui operations
Screen Snapshots	User captures ui screens
User Feedback	User submits comments
GOMS Analysis	Predict execution and learning time
UIDE Analysis	Conduct goms analysis within a uide
Cognitive Task Analysis	Predict usability problems
Task-Environment Analysis	Assess mapping of users goals into ui tasks
Knowledge Analysis	Predict learnability
Design Analysis	Assess design complexity
Programmable User Models	Write program that acts like a user
Information Proc. Modeling	Mimic user interaction
Petri Net Modeling	Mimic user interaction from usage data
Genetic Algorithm Modeling	Mimic novice user interaction
Information Scent Modeling	Mimic web site navigation

answer. Also, non-relevant factors are easily controllable. Lastly, due to the direct interaction, intentions and thoughts of participants can be properly examined.

When conducting a field study, researchers observe participants in their natural environment. It is less likely to present specifically designed tasks to users during such a study. Usually, one aims at letting users perform natural tasks. Participants interact with their environment as they normally would. Field studies are less controlled than lab studies (Dumais et al., 2014). However, it is still common for researchers to interact with users. Posing questions when participants are fulfilling tasks is essential to gain deeper insight into user behavior.

Log study is the third type of behavioral data collection, according to Dumais et al. (2014). It provides a way of observing natural user behavior uninfluenced by factors existing in the other two aforementioned study types. No observer is directly interacting with participants. Users operate their own equipment, mostly in a remote location. The usual downside of this approach is that barely anything is known about users and their intentions. Due to not being able to talk to participants, usage patterns have to be derived merely from log data. While this is a tedious task which leaves room for interpretation, it also has upsides. No one-on-one interaction is necessary. Therefore, collecting data scales much better in comparison to other approaches. Especially when a large number of samples should be investigated, log studies are ideal. Another point made by Dumais et al. (2014) is that actual, unsupervised behavior might differ from how participants would behave in the presence of an observer.

Grimes, Tang, and Russell (2007) take a different approach of categorizing types of data collection for UI evaluation. The authors of this work combine field study and lab study into one category. They also mention log study as another category. The main difference to the categorization described above is the mention of instrumented panels. Grimes et al. (2007) describe this concept as a periodic study with the same set of participants, using a specific program or browser application. Although participants have their actions recorded in a matter similar to log studies, it is also possible to collect qualitative data. This is done by questions defined beforehand. Those fixed questions are then posed to the participants directly in the browser. Vermeeren et al. (2010) also mention a supercategory similar to instrumented panels. However, they name it "online methods".

The categorization of data collection for UI evaluation methods established by Dumais et al. (2014) and Grimes et al. (2007) was merged and summarized in Table 2.5.

Furthermore, some researchers even propose elaborate taxonomies to differentiate between different UI evaluation methods in more granularity. Ivory and Hearst (2001) introduce four dimensions: method class, method type, automation type, and effort level. Other authors use a number of up to 18 characteristics to distinguish between methods (Vermeeren et al., 2010; Fernandez, Insfran, & Abrahão, 2011). However, going into detail about these taxonomies and characteristics would be beyond the scope of this work. Instead, the following section talks about the user interface evaluation method most relevant to this thesis, log analysis.

Table 2.5.: Categorization of UI evaluation methods based on definitions by Dumais, Jeffries, Russell, Tang, and Teevan (2014) and Grimes, Tang, and Russell (2007).

Type	**Level of Control**	**Level of Detail in Feedback**	**Tasks**	**Users**	**Environment**	**One-on-One**
Lab Study	High	High	Artificial	<50	Artificial	Yes
Field Study	Rather High	High	Arbitrary	<50	Natural	Yes
Interactive Panels	Rather Low	Rather Low	Natural	<1000	Natural	No
Log Study	Low	Low	Natural	∞	Natural	No

2.5.3. Log Study and Log Analysis

Data collected during log studies can be used for multiple types of analysis. Performance measurement (Agosti, Crivellari, & Di Nunzio, 2012), behavior model mining (Ghezzi, Pezzè, Sama, & Tamburrelli, 2014; Shin, Shafiei, Kim, Jain, & Raghavan, 2018), or other behavioral analysis (Busany & Maoz, 2016; Paganelli & Paternò, 2003) are just some examples of UI evaluation methods log studies render possible.

In the context of this thesis, performing log analysis to compare website versions is the most relevant use case. Utilizing this analysis, performance measurement is also possible. After data is collected, results must be analyzed. Then, a decision on which version is preferable can be made. For this, A/B testing and multivariate testing is discussed briefly.

Types of Web Logs

Basically, one has to distinguish between two types of Web logs: server-side and client-side.

Server-side logs are mostly captured automatically by server software, such as Apache Tomcat (The Apache Software Foundation, 2018c), and saved to log files. No modification of websites or installation of third-party software is necessary on user side. However, this only records very basic information about HTTP requests. Examples for such information are date and time, requested pages (URLs), and page size. It provides little support for including more information about users or general context. Utilizing dynamic websites might also lead to HTTP requests not being the main form of interaction with servers. Default server logs would lack information about basic user interaction. These concerns are also expressed by Fenstermacher and Ginsburg (2002), Hong, Heer, Waterson, and Landay (2001) as well as Ivory and Hearst (2001). Although server-side logs can be enhanced using techniques such as web beacons, they still lack rich information of client-side logs.

Logs generated on client-side can be tailored to contain more specific information. A definite downside of generating this log data is the effort necessary to monitor user activity. Realization of such observing mechanisms is much more complex than simple usage of server logs. They have to either be included in the browser, for example through browser widgets, or directly injected as part of website code.

Either way, incorporating frameworks to monitor user activity for web applications can be laborious.

Fenstermacher and Ginsburg (2002) outline four goals for client-side monitoring frameworks: no interference of user activity, user-individual data collection, possibility for developers to define which data is collected and how, as well as extensibility to other applications. Besides custom implementation of logging systems for specific use cases, existing frameworks and tools exist. WELFIT (de Santana & Baranauskas, 2015), WebQuilt (Hong et al., 2001), and WebRemUsine (Paganelli & Paternò, 2003) are three such tools mentioned in literature. Other examples of custom frameworks and tools were introduced by Fenstermacher and Ginsburg (2002), Vasconcelos, Santos, and Baldochi (2016) and Gerken, Bak, Jetter, Klinkhammer, and Reiterer (2008).

As Section 2.5.2 laid out, authors commonly disagree on the exact classification of user interface evaluation methods. However, many agree that utilizing log studies should go hand-in-hand with qualitative studies or interviews (Dumais et al., 2014; Grimes et al., 2007; Agosti et al., 2012; Vermeeren et al., 2010; Fernandez et al., 2011; Gerken et al., 2008). For example, according to Agosti et al. (2012) combining implicitly and explicitly collected data provides better understanding than each method on its own. They also note that *"logs alone give only a partial view of the stream of information that users produce"* (Agosti et al., 2012, p. 17).

Testing Different Versions

For testing of different website versions, A/B testing and multivariate testing is considered relevant. These methods for comparing performance of software features are explained well in literature, for example in Kohavi and Longbotham (2017), Olsson and Bosch (2014) and Kohavi et al. (2013). Such methods and statistical tests are applicable to many situations. However, websites are seen as the primary test subject in the elaborations below.

According to Kohavi and Longbotham (2017), in A/B testing users are evenly distributed into "buckets" (bucket test, split test). Each user is randomly put in a bucket. One user group is assigned a base version of the website (A). This version is usually called "control system" and holds no new features. The second user group is presented with an improved version of the current system when visiting the website (B). This version is commonly referred to as "treatment system". Figure 2.3 displays the basic idea of A/B testing according to Kohavi and Longbotham (2017). If possible, differences between versions should be kept as small as possible. This minimizes occurrence of false positives, side effects, and random factors. False positives refer to positive or desirable results which are not actually factual (Kohavi et al., 2013).

Multivariate testing is very similar to A/B testing. The only difference between these two methods is that multivariate testing compares multiple versions, while A/B testing only compares two. In some cases, multivariate testing might be referred to as A/B/n testing. Kohavi et al. (2013) claim that comparing only two

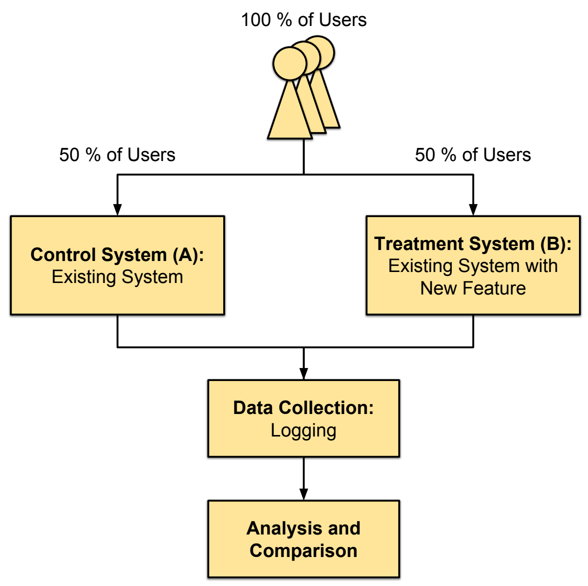

100 % of Users
50 % of Users
50 % of Users
Control System (A): Existing System
Treatment System (B): Existing System with New Feature
Data Collection: Logging
Analysis and Comparison

Figure 2.3.: Basic illustration of A/B testing. Adapted from Figure 2 by Kohavi and Longbotham (2017).

versions is preferable to comparing multiple ones. According to the authors, the main reason for this is that the risk of false positives occurring is higher in A/B/n than in A/B tests. Figure 2.4 shows a high-level illustration of the principle of multivariate testing.

There are two ways of conducting tests in the given context: experimental and observational (Kohavi et al., 2013). Experimental means that an controlled experiment is conducted. While one user group is presented with the control system, the other groups are assigned treatment systems. Observational data is collected when changes are introduced sequentially. Every user is working with the control system until the treatment system is deployed. Afterwards, analysis is performed. While multivariate tests naturally have to be conducted in an experimental way, A/B tests can be performed both experimentally and observationally. Sequential A/B tests have the upside of being better fitting for classic software release cycles. Therefore, they are easier to deploy. A negative factor is that they have an increased risk of higher false positive rates (Kohavi et al., 2013). Furthermore, order in which users are presented with different versions could change the outcome of tests. The overlying principle of sequential A/B testing can be seen in Figure 2.5.

For such tests, it is important to correctly define the metrics one wants to measure. However, there can be many rather insignificant ones. According to Kohavi and Longbotham (2017) it is essential to define an Overall Evaluation Criterion (OEC). This criterion should serve as a high-level metric for decision-making. Examples for an OEC are click-through rate, repeat user visits, or site performance.

After testing is finished, results have to be analyzed to determine the existence of a difference in performance. One possible approach is to derive the final result for

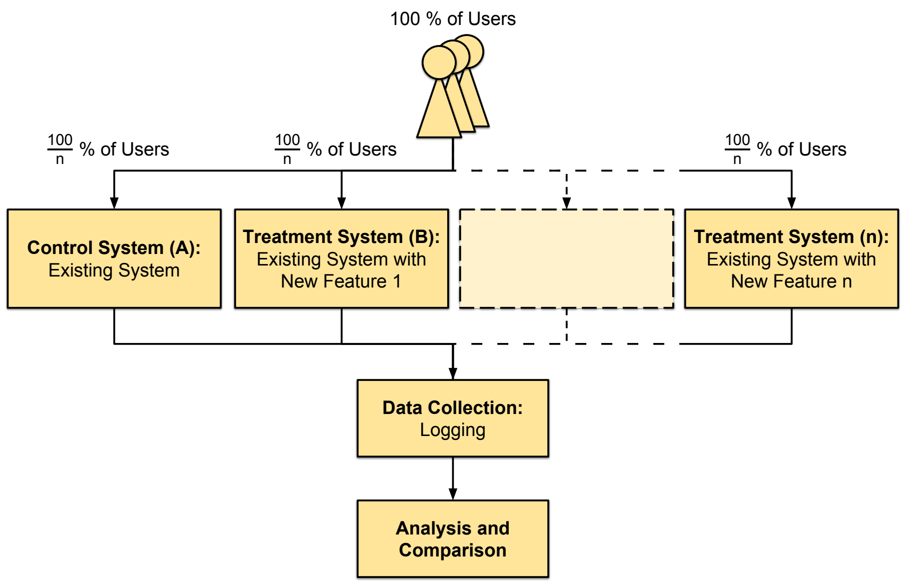

100 % of Users
100/n % of Users
100/n % of Users
100/n % of Users
Control System (A): Existing System
Treatment System (B): Existing System with New Feature 1
Treatment System (n): Existing System with New Feature n
Data Collection: Logging
Analysis and Comparison

Figure 2.4.: High-level illustration of the principle of multivariate testing (A/B/n testing).

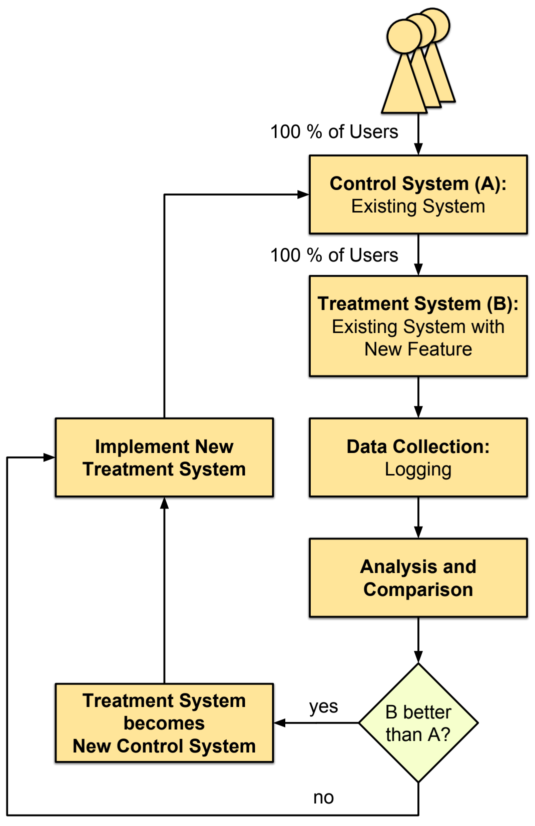

100 % of Users
Control System (A): Existing System
100 % of Users
Treatment System (B): Existing System with New Feature
Implement New Treatment System
Data Collection: Logging
Analysis and Comparison
Treatment System becomes New Control System
yes
B better than A?
no

Figure 2.5.: Basic concept of sequential A/B testing.

each metric from both versions' mean (Kohavi & Longbotham, 2017). In addition, statistical hypothesis testing can be used to determine if the findings are statistically significant (Kohavi & Longbotham, 2017; Kohavi et al., 2013; Defazio, 2016). Results would then be represented by a confidence interval. While details about statistical methods are not explained further, it should be mentioned that hypothesis testing can be performed using either a frequentist or bayesian approach (Defazio, 2016).

2.6. Summary

Government transparency and its correlated terms have surged up immensely in the past few years due to the technical possibilities the Internet provides. When legislative information is presented properly, websites make it easy for the general public to access proceedings and hold officials accountable for their actions. Up until now, many governments already implemented measures to enable open government and E-Democracy. However, not all governmental systems uphold the values of these principles. Numerous US state governments still shut the general public out of the lawmaking process. Therefore, need for citizen-instigated initiatives and platforms providing insight into legislative proceedings is still prevalent.

Before data can be presented to the public, preparations must be carried out. Automated pipelines preprocess data and information to enrich content. One of the most common applications for automatic procedures is the conversion of legislative hearing recordings into transcripts. These processes rely on technologies such as NLP, image recognition, and speech-to-text algorithms. In NLP, word similarity, sequence alignment, as well as NER- and POS-tagging are useful tools.

Since proceedings of political hearings contain crucial information, mistakes must be prevented whenever possible. Therefore, preprocessing is not usually performed in a completely automated way. Instead, systems frequently rely on human-assisted correction of automatically generated data. In addition to necessary error correction, humans annotate additional metadata to further enrich records. This is achieved by the use of annotation and transcription tools. Tools are available for many use cases, for example annotation of audio, video, or written data.

When discussing tool development, user interface design and evaluation is an important topic. Evaluation methods can be classified by how data is gathered: with lab studies, field studies, or log studies. Log analysis is especially interesting, since it enables extensive performance measurements in a real-world environment. It is also feasible to carry out long-term log analysis without significant disruption of the current workflow. To compare different versions of a system, log studies can be combined with A/B testing or multivariate testing. If the existing workflow should not be disrupted, multivariate testing is considered preferable. However, results could turn out to be inconclusive due to external factors and side effects.

3. Digital Democracy

Digital Democracy is a tool that lets you search for issues at the state level as easily as you might search for something in Google.

— CNN.com, *February 7, 2017*

There are many components contributing to the Digital Democracy initiative. The parts most relevant to this thesis are introduced in this chapter. To start, principles and history of the Digital Democracy initiative are covered. Afterward, the initiative's main tool for making information accessible, the Digital Democracy website, is presented. Then, the human-assisted pipeline utilized to process transcripts is explained. Lastly, functionality and user interface of the Digital Democracy Transcription Tool are examined.

3.1. Digital Democracy Initiative

The Digital Democracy initiative is a project instigated by the Institute of Advanced Technology and Public Policy (IATPP) at California Polytechnic State University (Cal Poly) in 2014 (California Polytechnic State University, 2014). As already shortly addressed in Chapter 1 and Section 2.2, the initiative aims at bringing transparency into the ongoings of US state legislature. This is achieved by developing a website which makes it easy for the general public to monitor legislative hearings and proceedings. On this website, fully searchable hearing transcripts are made available to users. As mentioned in Section 2.2, citizens of some US states would have no way of properly accessing and searching for information in their state's legislative proceedings otherwise. Lawmakers and lobbyists could not be associated with their own verbal remarks.

In 2015, Digital Democracy's online platform was first made accessible to the public. In an official press release issued by the Public Affairs Office of California Polytechnic State University (2015), former state senator Sam Blakeslee mentioned that Digital Democracy was developed to "open up government". He also stated that there are no transcripts produced by the California Legislature during state legislative hearings. He further explained that due to this fact the public would have no way of seeing what really happens in hearings.

Blakeslee's claim was backed by a poll conducted by the Institute for Advanced Technology & Public Policy (IATPP) at Cal Poly San Luis Obispo, which he founded. The results of this survey show that a large number of Californians would highly approve of changes to the legislature's transparency (Myers, 2015). Regardless of party or ideology, citizens demand more insight into politics. Most people find it especially important to have further access to information about budgeting. The interviewees were also very interested in being able to access documents and searchable information online (Myers, 2015).

Although Digital Democracy was initially supposed to only focus on California, other states are also lacking transparency regarding government proceedings. It soon became apparent that the ideas which led to founding this project are also applicable to other states in the USA. Therefore, the initiative integrated hearings held in New York into the system in early 2017. As of January 2018, the most recent additions to the selection of states for which Digital Democracy offers searchable information are Florida and Texas (Robertson, 2018). Digital Democracy therefore serves data for state legislations representing over 108 million people (United States Census Bureau, 2017).

Furthermore, there have already been talks with stakeholders from Colorado, Michigan, Nebraska, and North Carolina to further expand the system into these states (Robertson, 2018). However, due to an ever-growing amount of data and the need to keep costs as low as possible, this is a difficult task. At the time of writing this thesis, scaling the system to cover these states would be an immense challenge. In addition to scaling issues, some target states assert copyright protection over its videos of legislative hearings, such as Illinois (Robertson, 2018). This prevents public reuse of these proceedings and further showcases the need for a platform like Digital Democracy.

Section 3.2 below introduces the Digital Democracy website. However, presentation of information on this platform is only a fraction of what the project actually achieves. Before data can be queried and browsed properly, many preprocessing steps must be taken to provide correct information to users of the service. Essentially, transcripts have to be generated from videos of legislative hearings and floor sessions, since most states do not create any written records of hearings. Because of this, the initiative has technical issues to tackle and tasks that need to be solved. For example, correctly transcribing videos of hearings and detecting which person is currently speaking is one of these challenging tasks. Because of the difficulties caused by this process, the Digital Democracy initiative uses a human-assisted transcription process with two separate phases (Rovin, 2016). While Figure 3.1 shows a simplified overview of this pipeline, Section 3.3 elaborates on it in more detail.

3.2. Digital Democracy Website

The website provided by Digital Democracy (www.DigitalDemocracy.org) can be described as a statehouse accountability platform (Digital Democracy initiative, 2017a). It creates a searchable, verbatim record of all statements, whether they were made by lawmakers or witnesses during hearings in legislative committees

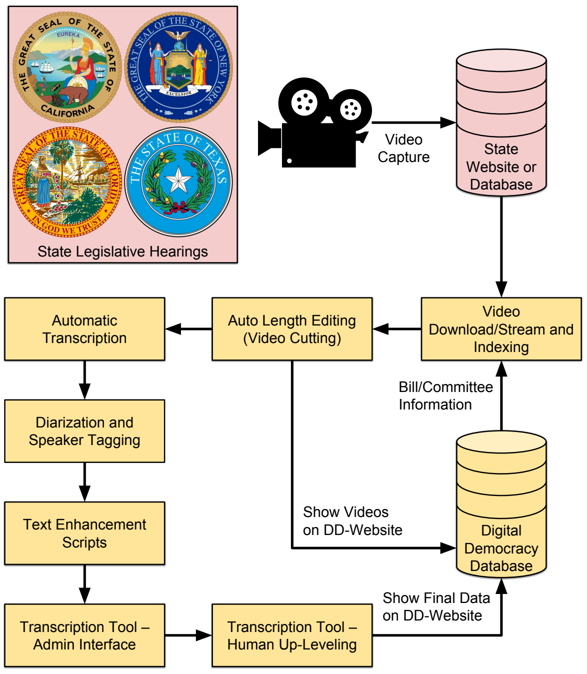

State Legislative Hearings
Video Capture
State Website or Database
Video Download/Stream and Indexing
Auto Length Editing (Video Cutting)
Automatic Transcription
Diarization and Speaker Tagging
Text Enhancement Scripts
Transcription Tool – Admin Interface
Transcription Tool – Human Up-Leveling
Bill/Committee Information
Digital Democracy Database
Show Videos on DD-Website
Show Final Data on DD-Website

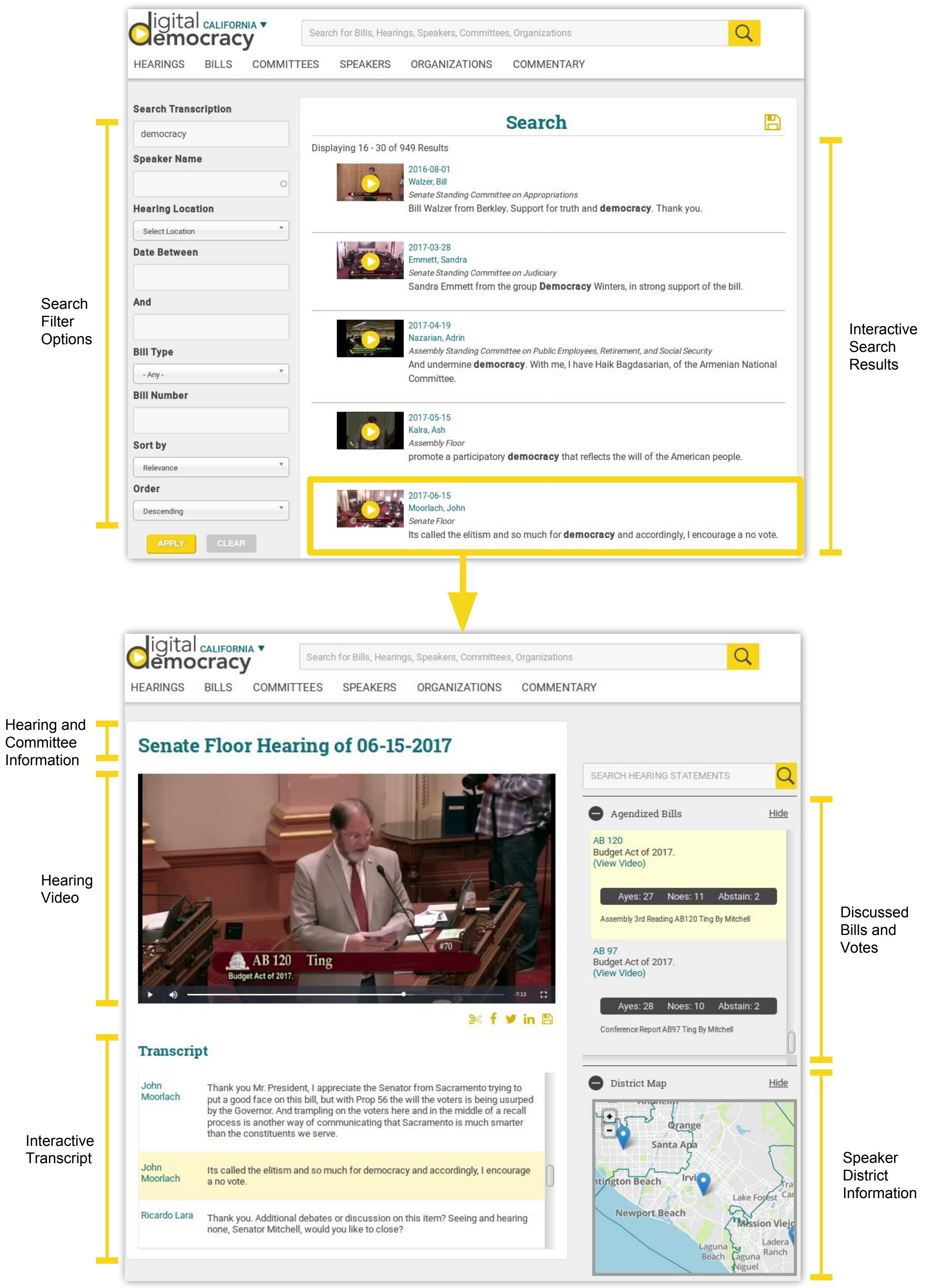
Figure 3.1.: Simplified view of the Digital Democracy processing pipeline. Adapted from Figure 1 by Khosmood, Dekhtyar, Assai, Kurfess, and Snyder (2014).

and floor sessions in statehouses. The website opens up legislative proceedings to everyone. It provides a simple interface to query for specific information in the full-text of transcripts for all users.

After a successful search request, a list of results is provided. Each entry in the result list holds the part of the transcription text in which the search term occurred as well as a direct link to the hearing where it was mentioned. The link takes the user to the web page of the specific hearing and sets the video to the exact position where the search term was mentioned. Figure 3.2 shows an example of how Digital Democracy displays search results and links transcription as well as metadata to the actual hearing video.

While this simple search interface is the centerpiece of the Digital Democracy website, one is also able to find additional information about state government there. Users can browse through information about hearings, bills, committees, speakers, as well as organizations and lobbyists.

The Digital Democracy website is realized by usage of the content management system Drupal 8 (Drupal Association, 2018). Development, design, and implementation for this part of the initiative's system was not part of this thesis.



Search for Bills, Hearings, Speakers, Committees, Organizations

Search Transcription: democracy
Speaker Name
Hearing Location: Select Location
Date Between
And
Bill Type: - Any -
Bill Number
Sort by: Relevance
Order: Descending

Search

Displaying 16 - 30 of 949 Results

2016-08-01
Walzer, Bill
Senate Standing Committee on Appropriations
Bill Walzer from Berkley. Support for truth and **democracy**. Thank you.

2017-03-28
Emmett, Sandra
Senate Standing Committee on Judiciary
Sandra Emmett from the group **Democracy** Winters, in strong support of the bill.

2017-04-19
Nazarian, Adrin
Assembly Standing Committee on Public Employees, Retirement, and Social Security
And undermine **democracy**. With me, I have Haik Bagdasarian, of the Armenian National Committee.

2017-05-15
Kalra, Ash
Assembly Floor
promote a participatory **democracy** that reflects the will of the American people.

2017-06-15
Moorlach, John
Senate Floor
Its called the elitism and so much for **democracy** and accordingly, I encourage a no vote.

Senate Floor Hearing of 06-15-2017

SEARCH HEARING STATEMENTS

Agendized Bills

AB 120
Budget Act of 2017.
(View Video)
Ayes: 27 Noes: 11 Abstain: 2
Assembly 3rd Reading AB120 Ting By Mitchell

AB 97
Budget Act of 2017.
(View Video)
Ayes: 28 Noes: 10 Abstain: 2
Conference Report AB97 Ting By Mitchell

District Map

Transcript

John Moorlach	Thank you Mr. President, I appreciate the Senator from Sacramento trying to put a good face on this bill, but with Prop 56 the will the voters is being usurped by the Governor. And trampling on the voters here and in the middle of a recall process is another way of communicating that Sacramento is much smarter than the constituents we serve.
John Moorlach	Its called the elitism and so much for democracy and accordingly, I encourage a no vote.
Ricardo Lara	Thank you. Additional debates or discussion on this item? Seeing and hearing none, Senator Mitchell, would you like to close?

Figure 3.2.: Search functionality of the Digital Democracy website as well as transcription and metadata display.

3.3. Human-Assisted Transcription Pipeline

Generating data and providing information for the Digital Democracy initiative is a two-phase process. First, an automated data processing chain utilizing an automatic speech recognition service transcribes the video of a hearing. However, this process of converting speech into text is error-prone. Furthermore, speakers cannot be easily identified automatically. Therefore, a second phase is needed in which humans manually correct errors of the speech recognition service. They also add annotations such as information about the current speaker. To ease manual correction, a web application called Transcription Tool was introduced. While the human-assisted pipeline is closer investigated in this section, Transcription Tool is explained in more detail in Section 3.4.

Even outside the pipeline introduced here, automatic scripts add records to the Digital Democracy Database (DDDB). Depending on state, year, and committee specific scripts pull data from third-party sites. That metadata is then incorporated into the DDDB and connected to the rest of the hearing data. Since this thesis mainly focuses on the improvements made to the Transcription Tool, not much detail will be given about these exact data sources. For example, additional information for California hearings and the people speaking in these discussions is extracted from MapLight, the website of the California Secretary of State, and The California Channel (Khosmood et al., 2014; Rovin, 2016).

Figure 3.3 visualizes the human-assisted transcription process as an activity diagram. Most of the automated parts of the pipeline visible in this figure are beyond the scope of this thesis. However, it may be important for the reader to understand where and when the information and metadata transcribers use as a starting point for their work are generated. Therefore, the following few paragraphs describe Digital Democracy's human-assisted approach in more detail. This diagram is modeled after the state of the preprocessing pipeline in spring 2018. As already mentioned, Digital Democracy is evolving. This leads to parts of the project such as Transcription Tool and its task generation logic, the preprocessing pipeline, or auto-correction scripts constantly being updated.

The processing pipeline uses hearing videos which were either recorded from a live stream or downloaded from a video archive as an input. The actual video source is dependent on the state. After the video was successfully stored and indexed in the Digital Democracy video archive, automatic trimming and cutting is performed. This shortening and partitioning of videos into clips serves multiple purposes. First, recordings of hearings might contain silent periods at the beginning and end of a video, which have to be removed. Second, videos are cut into separate clips with a length of about thirty minutes. This guarantees faster generation of automated transcripts by the external transcription service.

When trimming and cutting is finished, the hearing appears in the admin interface of the Transcription Tool. Admins can then review the clips and make adjustments by manually trimming or cutting it. If everything is in order, they send the video to an external transcription service via the user interface. Cielo24 (cielo24, 2018) is the

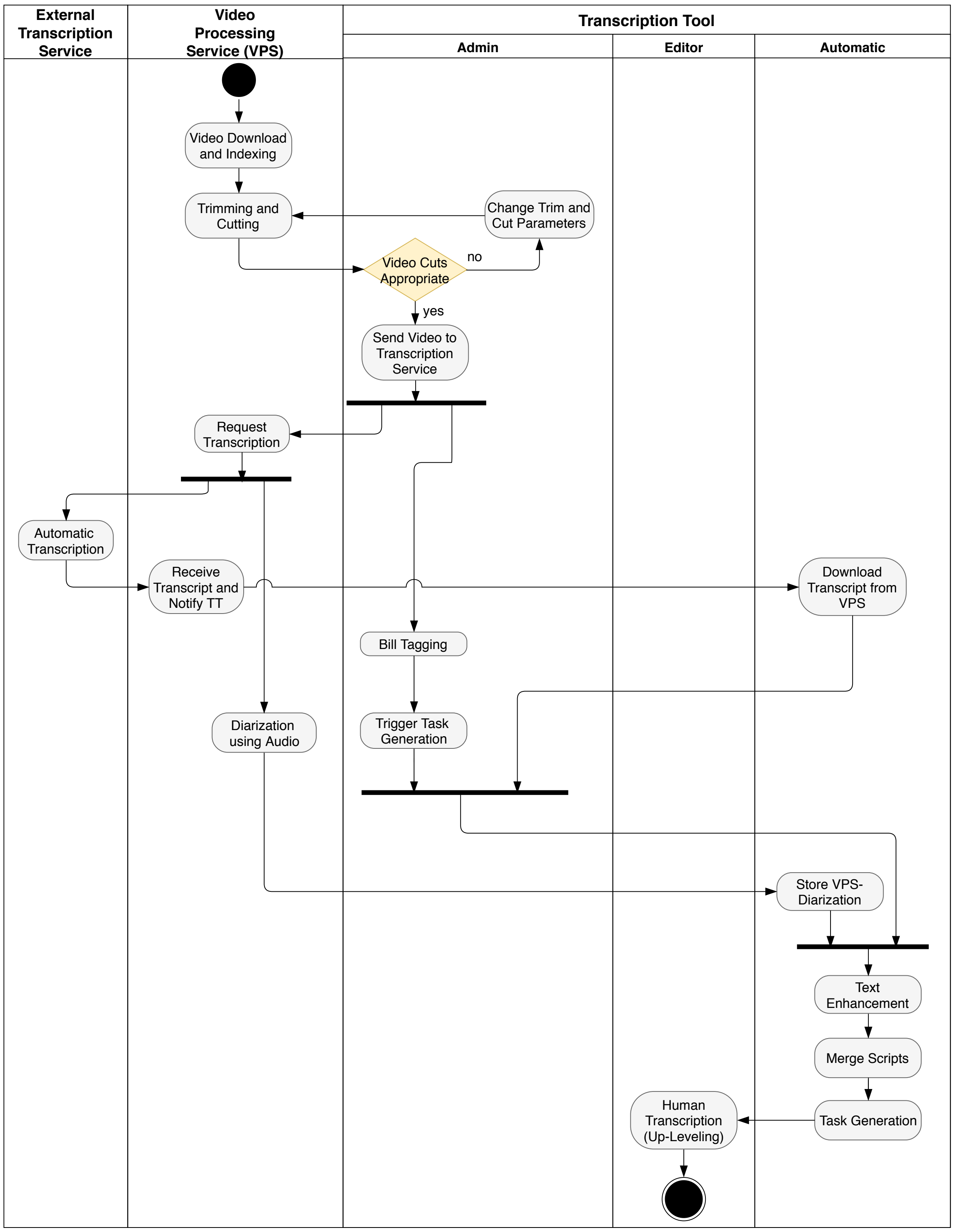


Figure 3.3.: Activity diagram of the human-assisted transcription pipeline.

currently used service to generate automated transcripts. In the near future, Digital Democracy plans to also enable usage of other engines such as Watson (IBM, 2018) as possible transcription services. The video transcripts returned by the external services are stored in SubRip subtitle (SRT) files. Each file holds the full transcript of a single video, fragmented into short intervals of speech, called utterances. In the context of Digital Democracy, an utterance is defined as one to many sentences spoken by the same person containing a line of thought.

Submitting a hearing video to an external transcription service also triggers a diarization process using audio and text. For this diarization of utterances a toolbox introduced by Rouvier, Gay, Khoury, Merlin, and Meignier (2013), called "LIUM_SpkDiarization", is currently used. Speaker diarization describes the process of identifying which people spoke during which intervals, without knowledge of their exact identity. Speech data such as parts of audio streams or segments of texts can then be tagged and linked to an unidentified person. It is also possible to identify people via facial or lexical features. Digital Democracy only performs diarization by audio and text, although future projects aim at also including visual clues such as facial features (Kauffman, Williams, et al., 2018). During this process, speaker tags (diarization tags) are assigned to separate utterances. They are then utilized to determine when speaker changes occur and which person spoke when. Each of the generated tags represents a different, unidentified speaker. Producing this information is one of the most crucial steps, since work done by human transcribers is much easier (and therefore faster) with correct speaker tags.

After sending the video to the external service, administrators add more information to the hearing using the Transcription Tool. For example, they annotate the hearing with committee and bill discussion information. Adding this information is called "Bill Tagging". In every hearing, multiple bills could be discussed over different or overlapping periods of time. Creating this data is essential to provide proper information and correct data for the Digital Democracy initiative as a whole.

As one of the final steps of the preprocessing pipeline, automatic text correction is executed on the raw transcript. Python scripts remove consecutively occurring white spaces, capitalize proper nouns, and convert lexical representations of numbers to numerical ones (Rovin, 2016). If needed, utterances which are too short to stand alone are merged into longer ones. After this automatic preprocessing of transcripts has finished, segments of a hearing video for which automatically generated transcripts are already available are assigned to transcribers. Then, transcribers correct and enhance transcription data. These segments of video are called transcription tasks and are automatically generated by the tool, making use of the previously entered hearing information (Rovin, 2016). A task represents a short work package which is assigned to a single transcriber. Administrators can choose to either manually assign specific transcribers to important tasks, or let the Transcription Tool automatically distribute the workload among them.

The time span and content of tasks generated out of a hearing depend on the nature of the hearing as well as the aforementioned information added by an admin. For the sake of completeness, the correlation between hearing, hearing videos, bill discussions, and tasks is shortly explained. First, a hearing can be made up of multiple hearing videos. Hearing videos are also split up into multiple shorter videos (about 30 minutes) to ensure faster processing by the automatic transcription service. Over the course of a hearing, multiple bills can be discussed. Bill discussions can continue over different video fragments, while multiple bills can also be discussed at the same time. Tasks are then generated in such a way that there is one full bill discussion per task. If bill discussions overlap multiple videos or multiple bills are discussed at once, a more sophisticated approach is used. However, describing this approach here would be too much of a technical detail.

Finally, human transcribers start to enhance the preprocessed transcripts (upleveling) by working on the tasks assigned to them. Transcribers do this by using a software called Transcription Tool, elaborated in Section 3.4 below.

3.4. Digital Democracy Transcription Tool

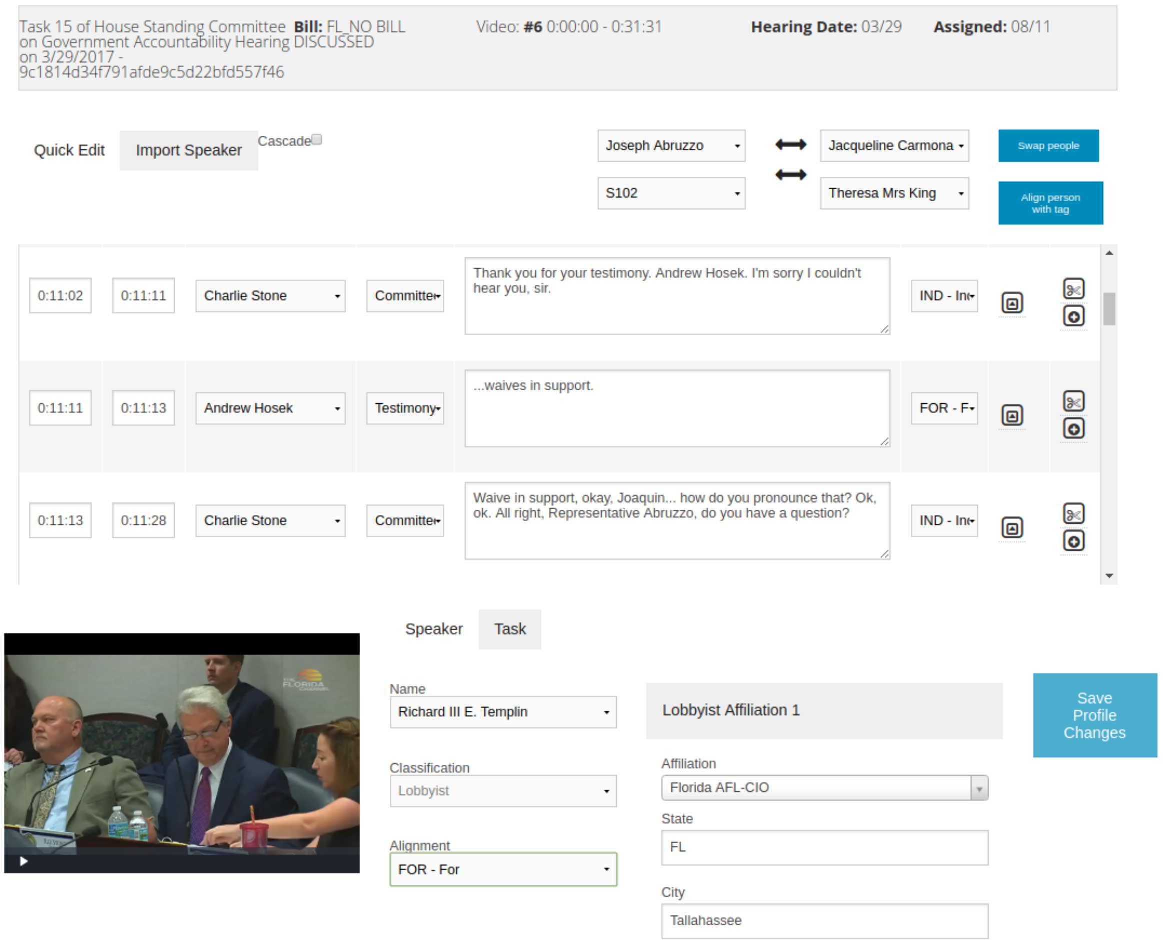
3.4.1. General Functionality

The Digital Democracy Transcription Tool serves as the main source of semi-structured data for most of the information and content provided by the Digital Democracy initiative and its website (Rovin, 2016). It is developed in-house. One of the tool's purposes is to handle administrative tasks such as importing hearing videos, updating hearing metadata, or supervising the transcription process. Despite the importance of its administrative features, Transcription Tool is mainly used by human transcribers to edit automatically generated transcripts in a distributed, asynchronous way, allowing for workforce flexibility. Due to these different usage scenarios, two user roles exist: "Admin" and "Editor". Admins handle administrative tasks while also being able to edit any transcript they want. All transcribers have the "Editor" role, which limits them to only access transcripts which are specifically assigned to them.

Transcribers enhance the previously automatically generated transcripts by using the tool's transcription user interface. Part of the transcription screen can be seen in Figure 3.4.

Administrators are mostly staff working for the IATPP. Transcribers are student workers employed on a short-term basis. Most students are initially not familiar with the terminology used in a legislative setting as well as the transcription interface itself. Due to the relatively high turnover of student staff and the cost of training new transcribers, it is necessary for the tool to work efficiently and provide an easy and straightforward interface. Transcriber numbers fluctuate over time. In summer 2017, five transcribers were working full-time with the tool. However, 36 transcribers and four admin users were interacting with the tool on a regular basis in spring 2018, mostly part-time.

The necessity for human transcribers stems from the fact that the textual tran-



Task 15 of House Standing Committee **Bill:** FL_NO BILL on Government Accountability Hearing DISCUSSED on 3/29/2017 - 9c1814d34f791afde9c5d22bfd557f46 Video: **#6** 0:00:00 - 0:31:31 **Hearing Date:** 03/29 **Assigned:** 08/11

Joseph Abruzzo ⟷ Jacqueline Carmona
S102 ⟷ Theresa Mrs King

0:11:02	0:11:11	Charlie Stone	Committee	Thank you for your testimony. Andrew Hosek. I'm sorry I couldn't hear you, sir.	IND - Ind
0:11:11	0:11:13	Andrew Hosek	Testimony	...waives in support.	FOR - F
0:11:13	0:11:28	Charlie Stone	Committee	Waive in support, okay, Joaquin... how do you pronounce that? Ok, ok. All right, Representative Abruzzo, do you have a question?	IND - Ind

Name: Richard III E. Templin
Classification: Lobbyist
Alignment: FOR - For

Lobbyist Affiliation 1
Affiliation: Florida AFL-CIO
State: FL
City: Tallahassee

Figure 3.4.: Transcript editing interface in the Transcription Tool as of October 2017.

scripts produced by automatic systems are not high quality enough for professional and government purposes. In such settings, even slight inaccuracies are highly problematic. Especially legislative bill information and personal names are not always correct. To fulfill the professional requirements demanded for proceedings of legislative hearings, these mistakes have to be corrected. Additionally, transcribers must identify speakers and decide on their alignment regarding the current issue discussed in the hearing. Lastly, they work to standardize utterance length by merging or splitting utterances. This allows for proper presentation of transcripts on the Digital Democracy website. Transcribers have to make sure no utterances exist which are too short to stand alone.

3.4.2. Transcription Screen User Interface

The Transcription Tool provides multiple screens and interfaces for both admins and transcribers. However, only the screen most relevant to this thesis is explained in more detail here – the transcription screen. Its interface is separated into three areas, which are explained from top to bottom of the web page.

Task Information Header

First, information about a task is displayed in a header. It provides contextual data about a transcriber's current work assignment. Furthermore, this information is useful to software developers for gathering information during debugging. An example of this informational header can be seen at the top of Figure 3.5.

The following fields are contained in the header:

- **Task Name and FileId**

  Name of the task, made up of this task's number in the current hearing, the hearing and committee name, as well as the hearing date (for example "Task 7 of Senate Standing Committee on Education Hearing on 5/18/2018"). For debugging reasons, the file ID for the video cut is also displayed.
- **Bill**

  Unique identifier of the bill, especially useful to transcribers. Using this data, they can look up further information about this bill on state websites. The bill identifier is a combination of state name abbreviation, session year, numbered revision of the bill, and official name of the bill (for example "TX_20170HB3632"). If no bill is discussed in this video, everything but the state name abbreviation is removed and replaced by "NO_BILL_DISCUSSED".
- **Video**

  Number and time interval of the video in the series of videos that make up this hearing.
- **Hearing Date**

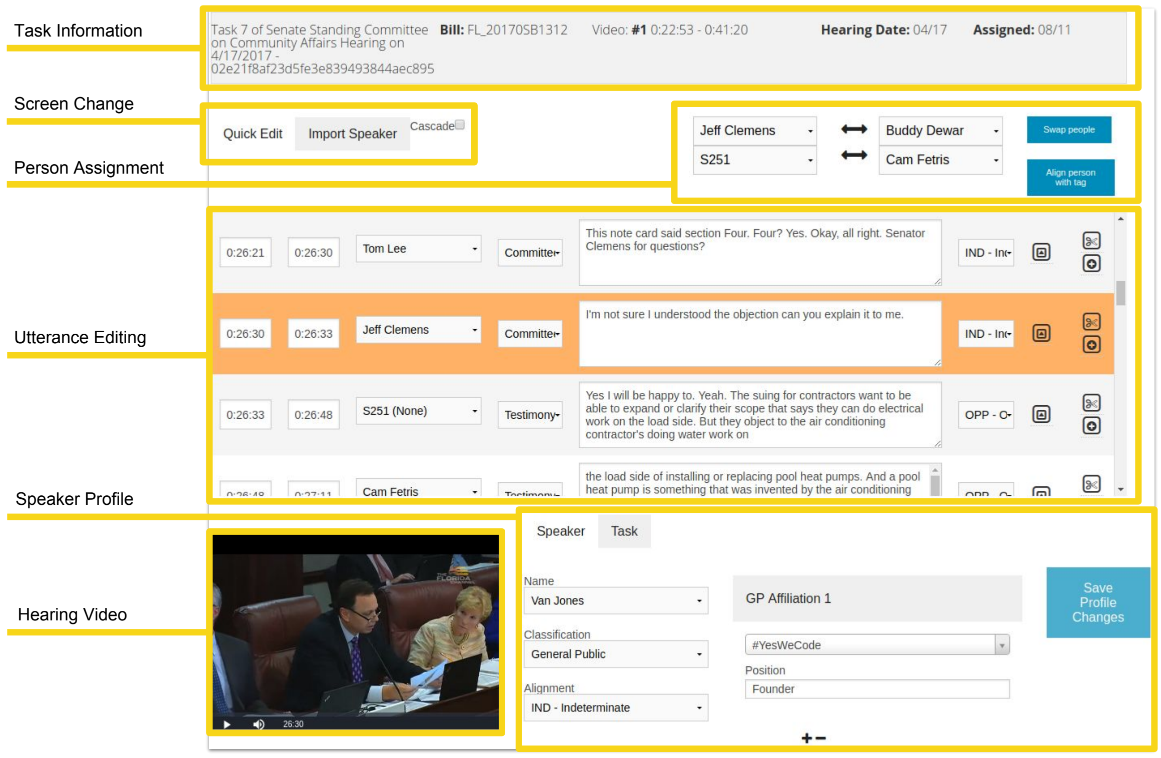
  The date the hearing was held.
- **Assigned**

  The date this task was assigned to a transcriber.

Utterance Editing and Speaker Import

The center part of the transcription interface is organized in two tabs. One screen holds all information about utterances existent in this transcript (Utterance Editing Screen). The other screen contains interface elements for importing speaker data into this task (Import Speaker Screen). Only after importing this data, transcribers can assign speakers to utterances.

To the right of the tabs which allow the user to switch screens, the person assignment interface is located. These elements enable transcribers to either set one person to all utterances of another one ("Swap people"), or assign a person to



Task 7 of Senate Standing Committee on Community Affairs Hearing on 4/17/2017 - 02e21f8af23d5fe3e839493844aec895 **Bill:** FL_20170SB1312 Video: **#1** 0:22:53 - 0:41:20 **Hearing Date:** 04/17 **Assigned:** 08/11

Jeff Clemens	Buddy Dewar
S251	Cam Fetris

0:26:21	0:26:30	Tom Lee	Committe	This note card said section Four. Four? Yes. Okay, all right. Senator Clemens for questions?	IND - Inc
0:26:30	0:26:33	Jeff Clemens	Committe	I'm not sure I understood the objection can you explain it to me.	IND - Inc
0:26:33	0:26:48	S251 (None)	Testimony	Yes I will be happy to. Yeah. The suing for contractors want to be able to expand or clarify their scope that says they can do electrical work on the load side. But they object to the air conditioning contractor's doing water work on	OPP - O
		Cam Fetris		the load side of installing or replacing pool heat pumps. And a pool heat pump is something that was invented by the air conditioning	

Name: Van Jones
Classification: General Public
Alignment: IND - Indeterminate
GP Affiliation 1
#YesWeCode
Position: Founder

Figure 3.5.: Description of elements in the Utterance Editing Screen of the Transcription Tool.

all utterances of a specific diarization tag ("Align person with tag"). The interface elements described here can be seen in the upper part of Figure 3.5 ("Person Assignment").

The Utterance Editing Screen is the most important part of the transcription interface. It allows transcribers to modify the automatically generated transcript. Whenever a transcriber edits data, changes are batched and sent to the transcription server in a 30 second interval. An overview of the utterance list is presented in the center of Figure 3.5. The utterance currently played in the video is highlighted in orange. Figure 3.6 shows the utterance interface element in more detail.

The components of this element are:

- **Start and End Time**

  Start and end time of the utterance.

- **Speaker or Speaker Tag**

  Combo box used to assign a speaker to an utterance. The initial selection for an utterance speaker is the diarization tag (speaker tag) computed during preprocessing. If the diarization algorithm did not produce a tag for this utterance, "None" is shown.

- **Utterance Type**

Combo box for setting the type of an utterance. The utterance type determines if the contained utterance represents a testimony, author's presentation, or committee discussion.

- **Utterance Text**

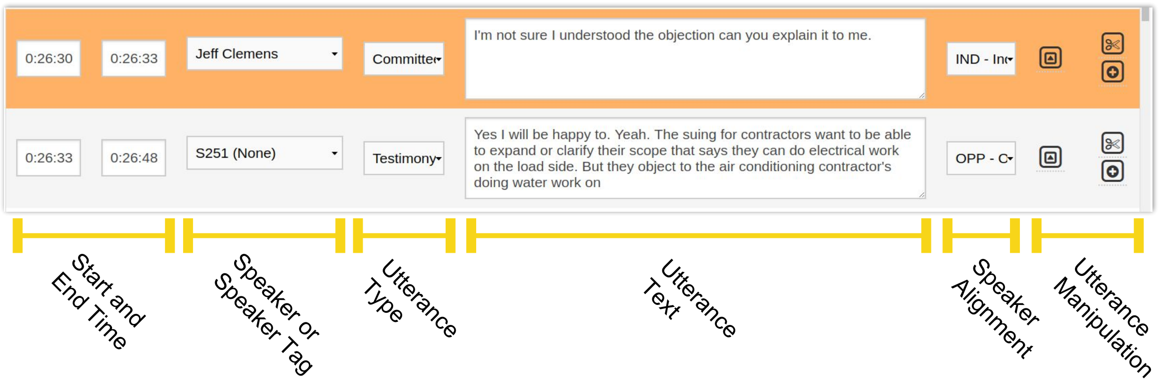
  The utterance text in an editable text area.

- **Speaker Alignment**

  Combo box to determine how a speaker's stance aligns with the currently discussed bill. Possible options for speaker alignment are: Indeterminate (IND), For (FOR), For If Amended (FIA), Neutral (NEU), Oppose If Amended (OIA), and Oppose (OPP). In practice, the most frequently used options are IND, FOR, and OPP.

- **Utterance Manipulation**

  These icon buttons enable merging and splitting of utterances as well as creating a new one. First, merging combines an utterance with the previous one. Start and end time are set according to the timestamps of the merged utterances. Besides that, utterance texts are merged. All other fields are set to the values of the previous utterance. Second, splitting separates an utterance at the current cursor position. The utterance text is split up, while all other data is copied for both utterances. Lastly, creating a new utterance leads to cloning of every field of the clicked utterance besides text. See Table 3.1 for further details on the utterance manipulation icons.



0:26:30	0:26:33	Jeff Clemens	Committee	I'm not sure I understood the objection can you explain it to me.	IND - Inc
0:26:33	0:26:48	S251 (None)	Testimony	Yes I will be happy to. Yeah. The suing for contractors want to be able to expand or clarify their scope that says they can do electrical work on the load side. But they object to the air conditioning contractor's doing water work on	OPP - C

Figure 3.6.: Interface element used to edit an utterance.

Table 3.1.: Interface icons for utterance manipulation.

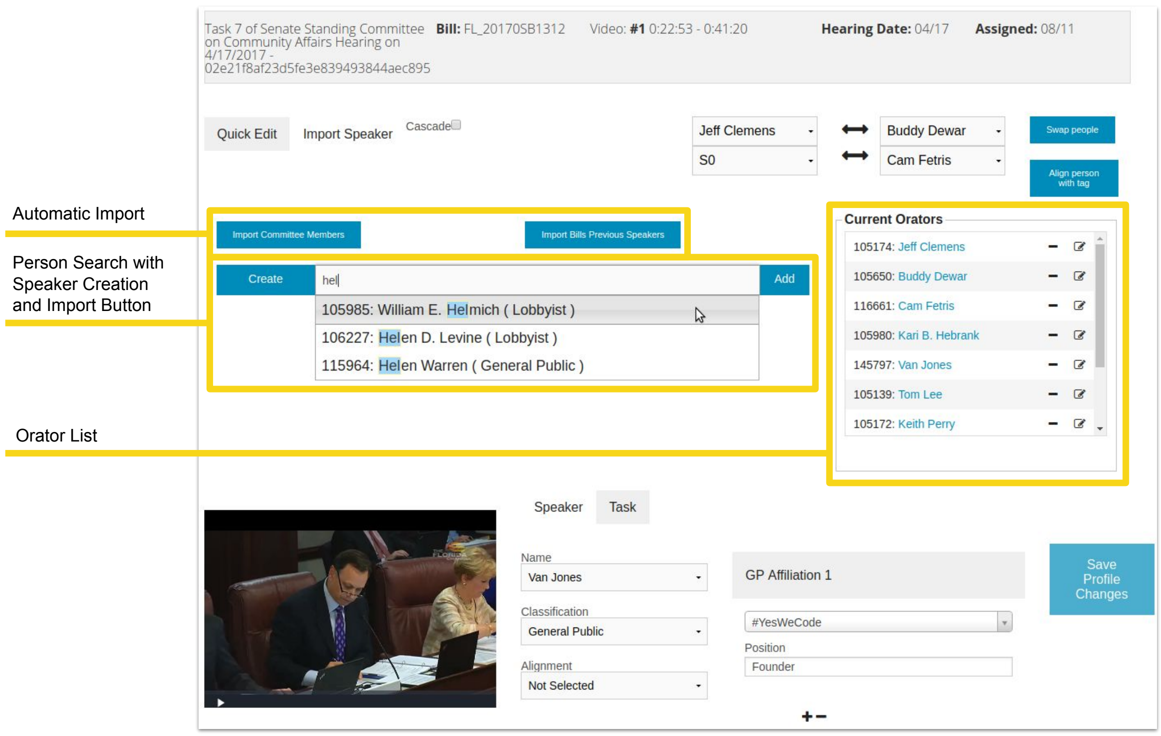
⊡	Merge utterance with previous utterance
✂	Split utterance at cursor position
⊕	Create new utterance by cloning existing one

As mentioned earlier, upon switching tabs the Import Speaker Screen is displayed in the center of the transcription screen. This collection of interface elements handles search, import, and removal of speakers for this task. Figure 3.7 shows this screen.

No person data is imported automatically, because the number of speakers in a single task is usually rather low. Instead, buttons exist which help mass-import data. “Import Committee Members” imports data about members of the committee holding this hearing. “Import Bills Previous Speakers” adds information about persons who previously testified in a hearing in which this bill (or one of its revisions) was discussed.

An auto-complete search box is located below the buttons enabling automatic import. New entries for persons can be created via a form opened using the “Create”-button. Clicking the “Add”-button adds the selected person information to the orator list (“Current Orators”) of the current hearing.

The “Current Orators”-list contains all persons whose data was previously imported into this task. Only persons in this list can be selected as speakers in utterances. This list also allows deletion of people from this task, if they are not assigned to any utterances. Clicking a speaker name opens that speaker’s profile on the Digital Democracy website in a new tab.



Task 7 of Senate Standing Committee on Community Affairs Hearing on 4/17/2017 - 02e21f8af23d5fe3e839493844aec895 **Bill:** FL_20170SB1312 Video: **#1** 0:22:53 - 0:41:20 **Hearing Date:** 04/17 **Assigned:** 08/11

Jeff Clemens ⟷ Buddy Dewar

S0 ⟷ Cam Fetris

hel

105985: William E. Helmich (Lobbyist)
106227: Helen D. Levine (Lobbyist)
115964: Helen Warren (General Public)

105174: Jeff Clemens
105650: Buddy Dewar
116661: Cam Fetris
105980: Kari B. Hebrank
145797: Van Jones
105139: Tom Lee
105172: Keith Perry

Name: Van Jones
Classification: General Public
Alignment: Not Selected
GP Affiliation 1
#YesWeCode
Position: Founder

Figure 3.7.: Description of elements in the Import Speaker Screen of the Transcription Tool.

Video Player, Speaker Profile, and Task Tab

The third part of the transcription screen is made up of the video player containing the hearing video and a tab widget containing two tabs. These tabs serve two purposes: modifying speaker profiles for this hearing ("Speaker", see bottom right of Figure 3.6) and task completion ("Task").

The speaker profile area is an important part of the transcription screen, since transcribers must select the correct classification of a speaker. Available speaker classifications are: "Unknown", "General Public", "Legislative Staff", "Legislative Analyst Officer", and "State Agency Representative". Although "Legislator" and "Lobbyist" also exist as valid classifications, speakers of these types can not be created manually. Instead, they are updated periodically by official data pulled from third-party sources. Depending on the classification, further data has to be added afterwards. For speakers of the general public and lobbyists, all organizations the speaker represents in this hearing should be added. State agency representatives and state constitutional officers must have their corresponding agency or office assigned to them. Information about which legislator they work for or which committee they serve has to be included for legislative staff. Legislative analyst officers is the only role for which no further specifications are necessary.

The last UI element in the speaker profile area is the alignment combo box. Choosing an alignment using this box automatically sets its value to all of this speaker's utterances. In the end, changes to a profile have to be confirmed using the "Save Profile Button".

The above-mentioned "Task"-tab contains interface elements relevant to task completion and error messages. For example, errors about missing speaker profiles or utterances without speaker selection are displayed here. If there are no error messages, the "Complete Task" button is shown. This button finalizes a task and stores all information that was not yet saved in the database. A button for reverting all changes that happened in the last sixty seconds is also present here.

3.5. Past Improvement Projects

Since Digital Democracy and its software components have been established as a well-working system, many improvements projects were attempted. Some of the more recent ones are described below, starting from the earlier adoptions up to the most recent one.

3.5.1. Video Caching

To establish better management of video clips and resources, Lam (2016) implemented a video caching system called Video Manager. This manager aims at diminishing necessary disk space on the live servers of the initiative. It also improves overall system performance, since videos do not have to be constantly downloaded from a cloud storage system.

Hearing videos are stored via the cloud storage system Amazon S3 (Amazon Web Services Inc., 2018). They are only loaded into the cache if they have to be accessed on the initiative's servers. An example for this is video cutting. Video Manager introduces an API for accessing videos. If a video is not in the cache, the manager downloads it from the cloud. The cache utilizes a Least Recently Used (LRU) algorithm. Lam developed Video Manager using Python and set up the server using its Web Server Gateway Interface (WSGI).

Although the cache implemented during this project increases scalability in general, future work might still be necessary. Lam explains that no tools for analysis were developed and no records of statistics are kept for now. Therefore, no possibility for measuring performance of the used caching algorithm exists.

3.5.2. Transcription Process Improvements

Rovin (2016) introduced improvements developed during separate projects into the main transcription process to increase efficiency. The three main changes integrated by Rovin were improvements to speaker diarization, text correction, and transcription UI. He also evaluated his changes to measure reduction of overall transcription cost.

Diarization Improvements

The changes to diarization implemented by Rovin (2016) were mainly directed at merging consecutive intervals (utterances) spoken by the same person. Usually, the automatic speaker diarization process would produce utterances too short to stand alone. Rovin developed Python scripts which would merge short, separate utterances into longer, more coherent ones.

Correction of Raw Transcriptions

Prior to the manual correction phase, Rovin integrated another automatic correction step. This change implements improvements regarding bill number correction and proper noun capitalization.

By default, automatic speech recognition might describe spoken numbers in lexical form. However, bill numbers in the state legislative system are usually depicted in numerical format. Numerical representation is also necessary to provide proper search functionality when querying for specific bills. Rovin integrated a process which converts specific lexical number representations to numerical ones. This is achieved by combining a machine learning classifier, parsing, and applying a grammar.

In addition, Rovin added text correction for capitalization of proper nouns to the transcription process. This process is automated using simple dictionary look-up.

The dictionary was created based on an online repository and is continuously expanded whenever new proper nouns occur.

UI Changes

After transcriptions for two legislative sessions were completed, Rovin collected transcriber feedback. Repetitive tasks and tedious actions mentioned by transcribers lead to development of UI improvements. As listed by Rovin (2016), Transcription Tool was enhanced through the following changes:

- Implemented functionality for directly importing committee members into the speaker list and added a commonly used placeholder for the "Committee Secretary".
- Added initialization of speaker profile information during import.
- Automatic assignment of "Alignment" and "Discussion Type" fields for concurrent utterances of the same speaker was implemented.
- All Legislators participating in discussions were automatically assigned the utterance type "Committee Discussion" while all other speaker classifications were initially set to "Testimony".
- Functionality to rewind a video for 5 seconds was set to the tilde key.
- Inserted a check box which enables the "Cascade" option. If this box is checked, information added to an utterance will also be assigned to the next nine utterances.
- Implemented functionality of automatically applying alignment status to all utterances of a non-legislator.
- Added a button to the utterance element used for splitting an utterance.
- Alphabetized content of the speaker drop down in the utterance interface as well as search results and list of orators.
- Specific conditions were introduced which prevent task completion until they are met.
- Hyperlinks directing to speaker profiles were added to speaker names.
- Increased utterance text area size to four rows.

Cost Function

Rovin defined a cost function to determine performance of the manual transcription phase performed by humans. Manual transcription cost was defined as the summarization of "EditingCost" and "MistakeCorrectionCost". While "EditingCost" straightforwardly describes the cost of initial editing which has to be performed in any case, "MistakeCorrectionCost" is more interesting for improvement evaluation according to the author. Rovin takes six commonly appearing mistake correction steps into account:

1. Re-opening the tool
2. Correcting misspelled words

3. Correcting utterance length
4. Correcting wrongly assigned speakers
5. Importing known speakers
6. Identifying and importing unknown speakers

Transcription Tool Versions for Evaluation

To evaluate if changes decreased the overall cost of the manual transcription phase performed by humans, Rovin provided four different Transcription Tool versions. Each version included a specific improvement to the one mentioned before. The versions used for Rovin's evaluation were:

1. **Baseline:** No improvements to the previous version.
2. **UI Improvements:** Improvements as described in section 3.5.2.
3. **Text Correction:** UI Improvements as described above as well as text correction features described in section 3.5.2.
4. **Diarization:** All improvements provided in tool version 2 and 3 as well as the diarization changes described in section 3.5.2.

Task Selection for Evaluation

The tasks Rovin presented to testers were selected to specifically address the given changes. Overall, 15 people participated in the experiment. Test users were split up into two classes: experienced transcribers (6) and beginners (9). Each user had to transcribe five videos. For this, the videos with parameters closest to average speaking duration and speaker count over all videos in the database were selected.

Test users were then further split up into four groups and they completed five transcription rounds. In each round, a task was transcribed via a separate tool version, while the first round was used for training. Tasks differed from group to group. In the beginning, transcribers used the baseline version of the tool. After each round, the tool was upgraded to a better version, as listed in Section 3.5.2.

Definition of Efficiency

Over the course of the past few sections, the terms efficiency and performance were used to describe how well the transcription process operates. Rovin split up measurement of transcriber efficiency into four metrics:

- **Task Duration:** The time it takes to complete one transcription task.
- **Text Alignment:** Overall number of words in a finished transcription which match the actual spoken content (control transcription). Rovin used a modified Needleman-Wunsch algorithm to measure this metric (see Section 2.3.2).
- **Speaker Alignment:** The percentage of correctly identified speakers (per name) over all utterances.
- **Utterance Length:** The percentage of utterances which were cut to an adequate length (7–385 words).

Results

Evaluation conducted by Rovin showed that changes significantly decreased task duration. Measured values of other metrics only increased insignificantly. Analysis was conducted by applying multiple regression as well as examining changes in averages.

Rovin found that his improvements reduced the average transcription time by 16.89%. The relative ratio of transcription to video time was reduced from 7:37 to 6:19. When attempting an evaluation of the total cost reduction, a decrease of 10.66% was measured. Assuming the labor cost to be $10 per hour, Rovin claims a cost reduction of $2.71 per video transcription.

3.5.3. Improvements to Transcription Tool Admin UI

Reinman (2016) improved the admin user interface for video cutting and adding meta information in the Transcription Tool. Some manual tasks were also automated. For this, Reinman separated the video editing page into two simplified pages. While the first screen is mainly used for cutting video clips, the second one allows administrators to edit meta information such as committees, bills discussed, hearing date, state, and task priority. After an admin finishes video modification and submits his changes, task creation is triggered.

3.6. Existing Need for Improvements

The Digital Democracy project is subject to constant new developments and improvement projects. Some of them were documented in Section 3.5 above. However, cost of both automatic as well as manual transcription processes are still tremendously high (Rovin, 2016; Robertson, 2018; Digital Democracy initiative, 2017b).

Especially the Transcription Tool operated by human transcribers poses a bottleneck for the whole initiative (Rovin, 2016). The software is still considered unstable by project authorities and considerable room for improvement exists. Due to addition of new states, processing overhead and necessary human labor is steadily increasing. Introducing new Transcription Tool improvements is therefore considered a main priority for the project's development team. This includes initiating side projects considering technologies such as speaker recognition, automatic detection of speaker alignments, and other possible enhancements (Kauffman, Williams, et al., 2018; Budhwar et al., 2018; Kauffman, Khosmood, Kuboi, & Dekhtyar, 2018). Especially the advances in multi-modal speaker recognition using voice, face, and text (VFT) analysis are eligible for inclusion (Kauffman, Williams, et al., 2018).

Furthermore, project authorities currently have no access to metrics measuring Transcription Tool efficiency and overall time needed to work on hearing transcriptions. Although Rovin (2016) attempted a controlled evaluation (see Section 3.5.2), there currently is no way of generating and monitoring authentic performance values. Supervisors are left in the dark considering transcriber efficiency in a real-world environment. Due to this information deficit, not even approximate future projections for workload are possible. Furthermore, this means that a considerable decrease in tool efficiency is not detectable right away. Such a decrease could for example stem from newly implemented software code introducing defects into the Transcription Tool, or a disruption in the automatic preprocessing pipeline.

Finally, the currently existing automatic preprocessing pipeline is not flawless. Simple NLP analysis could be carried out to gain insight into how common corrections performed by transcribers could be automated. Especially corrections of proper nouns (e.g. person and company names) could be executed automatically to further cut down transcription time. Looking into possible modifications might proof profitable, especially in the latter part of the preprocessing pipeline.

3.7. Summary

Digital Democracy is an initiative instigated by the IATPP at Cal Poly, San Luis Obispo. It aims at increasing transparency in state governments. The initiative's website gives citizens an ability to gain insight into state government proceedings of California, New York, Florida, and Texas.

Before hearing transcripts can be presented on the website, a human-assisted preprocessing pipeline must be executed to convert hearing videos into meaningful transcription data. This pipeline consists of two main phases. First, it utilizes automatic processes to generate raw transcription texts and basic metadata. Second, human transcribers enrich existing data by correcting texts, tagging speaker data, and selecting speaker alignments. In the end, data is saved to the database and citizens can access it using the Digital Democracy website.

The software developed in-house to perform this human transcription process is called Transcription Tool. It is mainly used by transcribers to work on transcription tasks. Furthermore, administrators utilize its admin interface to manage hearing data and assign tasks to transcribers.

Although Digital Democracy has been subject to multiple improvement projects over the years, demand for cost reduction still exists. Especially the bottleneck created by the human transcription process using the Transcription Tool is problematic. In addition, no automatic performance and efficiency evaluations of transcription time are currently possible. Furthermore, investigations into automatic text corrections could reveal possibilities for improvements to specific processing steps in the human-assisted pipeline.

4. System Design and Requirements

This chapter defines overall goals of the Digital Democracy Transcription Tool and objectives of this work. In addition, current Transcription Tool requirements and technology are examined closer. Afterward, new requirements necessary to reach the introduced goals are set.

All explanations are directed towards redesign and improvement of the Transcription Tool. Other system components of Digital Democracy are not subject to closer investigation, due to their irrelevance for this thesis.

4.1. Goals

Fundamental goals of the Digital Democracy initiative were already elaborated in great detail over the past few chapters. The whole purpose of the human-assisted processing pipeline laid out in Section 3.3 can be summarized as the approach of converting hearing videos into meaningful, rich transcripts which can then be presented on the website. Such transcripts do not simply consist of plain transcription text, but instead hold exact speaker tags and timestamps for every utterance. This structured data allows for proper indication of information and enables search mechanisms on the Digital Democracy website.

When talking about the Digital Democracy Transcription Tool, the overall goal can be named as providing a user-friendly interface to handle transcription processes and their administration. The tool should be as flawless and efficient as possible. However, this is currently not the case, as explained in Section 3.6. According to demands brought up in the previous chapter and requirements defined in this one, the following major goals can be laid out as the final aim of this work:

- Develop improvements for the Digital Democracy Transcription Tool to increase tool efficiency
- Implement a logging system which allows supervision of Transcription Tool performance and efficiency as well as general interaction patterns
- Evaluate tool improvements using the logging system
- Prove usability of logs for future research

In the remainder of this section, requirements as well as developments necessary to achieve these goals are defined elaborately.

4.2. Transcription Tool State of the Art

The Digital Democracy Transcription Tool has been in use since the initiative's founding in 2015. It has undergone some technology and specification changes in the past. This subsection lays out an overview of past tool requirement definitions and currently used technologies.

4.2.1. Requirements

During his improvement developments, Rovin (2016) defined two very general core requirement packages for Transcription Tool. While the first one specified automated processes and hearing import into the tool in more detail, the second one defined particular functionalities of the transcription process.

The following points summarize the general data processing and hearing import requirements as set by Rovin (2016):

- The tool facilitates the association and importation of hearing information into the database.
- Importing a new hearing requires hearing date, IDs of associated hearing videos, and corresponding pregenerated transcripts.
- The tool imports information about committees, bills, and bill discussions automatically.
- Administrators can associate committee, bill, and bill discussion information with a hearing.

Furthermore, transcription process specifications as of Rovin (2016) can be summarized as follows:

- The tool facilitates correction and annotation of hearing transcriptions.
- It provides an interface for users to modify and annotate words from transcription files.
- It allows users to: create and modify utterances, give utterances time intervals, and assign speakers to utterances.
- Modifications should be allowed on specific sections of raw transcriptions ("transcription tasks").
- Transcription task assignments should typically be 5 to 20 minutes long.
- Tasks are assigned to editors by administrators.
- Transcribers can use the tool to see all tasks assigned to them.
- Finished transcriptions are uploaded to the database by the tool and new transcription information is linked to corresponding hearing information.

While tool functionality has been expanded recently, these core requirements are still relevant. Requirements for new features are specified in later parts of this chapter.

4.2.2. Tool Technology

Transcription Tool is developed as a web application (Rovin, 2016). The backend uses the Java based web framework Stripes (Stripes, 2018). Program code is organized in a classic model, view, controller (MVC) architecture. All data is saved to the DDDB, which uses MySQL (Oracle Corporation, 2018). Like all Digital Democracy services, Transcription Tool and DDDB are hosted on an Amazon Web Services cloud server (Amazon Web Services, Inc., 2018). OrmLite is used as the object-relational mapping approach (Watson, 2017). HTML generated on server side is based on JSP (JavaServer Pages Technology, 2017). For the frontend, content is mainly provided by the template-driven JavaScript library Ractive.js (RactiveJS contributors, 2017), while also making use of native JavaScript and jQuery (The jQuery Foundation, 2018). The frontend component framework Foundation offers further utilities for proper website presentation (ZURB, Inc., 2018). Communication between client and server is handled via Ajax and JSON (Mozilla and individual contributors, 2018; JSON, 2018). Maven is used as the build tool for this project's code (The Apache Software Foundation, 2018d).

4.3. New Requirements and Technological Changes

The following few sections further specify requirements necessary to reach the goals defined in Section 4.1.

4.3.1. Rework of Original Transcription Tool

As mentioned in Section 4.2.2, initially Stripes was used for implementing server-side technologies. However, in spring 2017 the project authorities deemed this framework outdated and not well-maintained. Therefore, the first step of the practical work done during this thesis is to migrate the Transcription Tool from Stripes to the Spring framework (Pivotal Software, 2018c). During this, the current object-relational mapping approach implemented via OrmLite is replaced with capabilities Spring provides.

This is a major contribution which the author of this thesis carries out in collaboration with other members of the Digital Democracy project team. It also serves as a way of getting used to every component of the Transcription Tool and its code. Figure 4.1 displays current frameworks and technologies of the Transcription Tool, while also indicating which obsolete parts are going to be replaced.

The tool rework not only concerns components elementary to the transcription screen. Backend code for all functionality, including admin capabilities, has to be adapted.

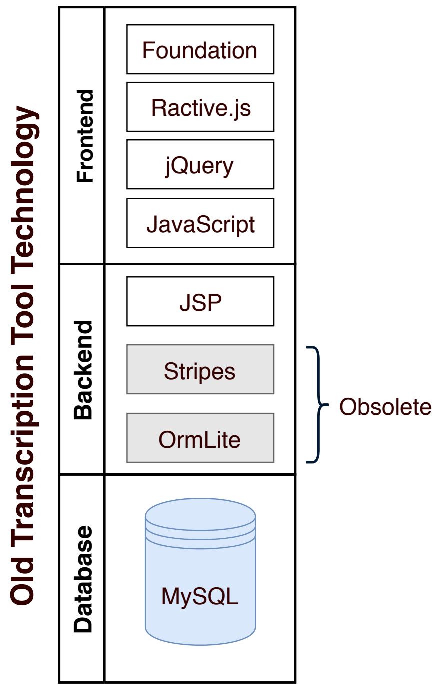


Figure 4.1.: Old Transcription Tool technology. Gray boxes represent obsolete components subject to replacement.

4.3.2. Data Collection and Logging System

While some students are always working from a dedicated workspace in an office, others work remotely. Because of this and the students' unregulated working hours due to them also being occupied with their studies, it is difficult to monitor the current progress of transcription tasks. Without keeping track of tool interactions, no simple way of properly quantifying transcriber productivity and therefore tool efficiency exists. There is no possibility of estimating workload produced by incoming transcription tasks.

Therefore, a logging framework must be introduced to enable performance measurement and detailed analysis. Client-side logs are used for this, because they allow for collection of richer data (see Section 2.5.3). Implementation should be carried out in JavaScript and jQuery to prevent overhead and ensure simplicity. Collected log entries are transferred to the server through the Transcription Tool. To prevent sending an excessive amount of HTTP requests, log entries should be collected and sent as batches. Preferred format for structuring and storing logs is JSON. Working on transcription tasks will produce a separate log file for each task.

Furthermore, automatic Python scripts must be built which insert log entries stored in JSON-files into the database. Preferably, these daily scripts are run overnight to not disrupt transcribers' standard working times. Figure 4.2 shows a general overview of this data collection approach.

Without looking at the technical realization in too much detail, at least the following interactions between user and tool should be captured by logs:

- Opening/Closing the transcription screen
- Editing utterance texts
- Changing speaker assignments
- Clicking utterance manipulation buttons (split/merge/new)
- Video player interaction
- Leaving the website/browser tab and coming back
- Importing, searching, and removing speakers
- Completing tasks
- Keyboard shortcuts

For each of these interactions, the following information should be collected:

- Descriptor/Name of interaction or event
- Exact time the interaction occurred
- Type of website element that user interacted with
- Unique identifier of website element that user interacted with
- Content of element before/after transcriber interacted with it
- Pressed keys or type of mouse click triggering the interaction
- Transcriber who interacted with website
- IP address or identifier to determine if interaction was done in project lab or from home
- Tool version transcriber worked with

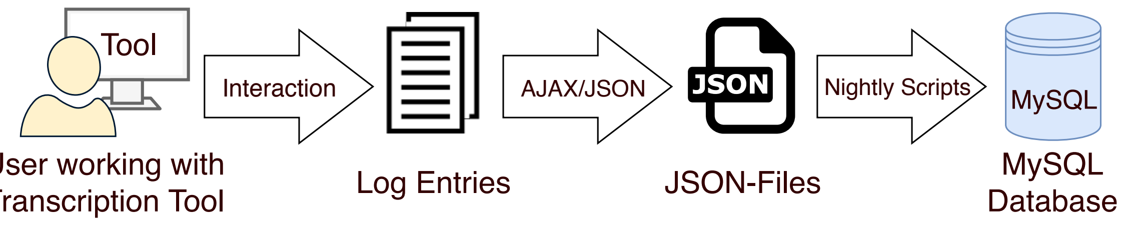


Figure 4.2.: Data collection pipeline for the Transcription Tool logging system.

4.3.3. New Feature Definitions

After finalizing the first rework and log system developments, new features which improve performance and efficiency of the transcription process should be implemented. For this, current problems and possible improvements to Transcription Tool had to be identified.

This was done by interviewing transcribers as well as talking to administrators. Interviewing users about basic needs was a requirement set by the author. However, it was decided that in-detail user interviews are not relevant for performance evaluation within the scope of this work. Interviews served two specific purposes:

Understanding current tool issues and making sure no feature is implemented which would not fit future transcriber needs.

For this, transcribers were first asked which part of working on a transcription poses the most effort for them. Second, they were questioned if they could think of any improvements for the tool. Then, results of the conducted interviews were combined with suggestions made by the project leaders.

Determining Transcriber Needs

In September 2017, five transcribers worked full time for Digital Democracy. An interview with each of them was conducted, evaluating the old version of the Transcription Tool.

Effort necessary to identify speakers was named to be one of the main factors increasing transcription time. In addition, splitting up utterances to achieve correct length and speaker assignments was identified as an expensive factor. In many cases, a textual utterance which the diarization scripts determined to be spoken by one person actually contained sentences by another speaker. In general, utterances seemed to be too long. The video player was also lacking options, such as enlarging the video or other more convenient ways to interact with it. Another request mentioned by the transcribers was that names of speakers in the database could not be changed via the Transcription Tool. Furthermore, transcribers unanimously agreed that correcting text is not a major effort. Although names have to be corrected frequently, ordinary text is generally of high quality. Lastly, transcribers criticized the long loading time of the tool.

Feature Packages

Based on suggestions by transcribers and project leaders, several possible changes aiming at improving stability, usability, and efficiency of the Transcription Tool were found. The following feature packages are deemed first-priority and are chosen to be implemented:

- **Profile Preview**

  A speaker profile picture preview next to speaker names should be added. This simplifies identifying people and enhances the search interface of the Transcription Tool.

- **Video Features**

  Better video player functionalities such as full-screen mode as well as slowing down and speeding up video are implemented. While in full-screen, it should be possible for transcribers to interact with other browser tabs.

- **Utterance Navigation**

Due to existing difficulties when navigating a transcript and the corresponding video, additional interface buttons are added. These buttons set the current video time to the beginning of utterances. Jumping to the previous or next utterance from the currently played one should also be enabled using additional UI elements. Lastly, error messages displayed in the "Task"-tab are made interactive.

- **Speaker Recognition using VFT**

  Other student projects and theses focus on providing better speaker recognition while combining voice, face, and text analysis (Kauffman, Williams, et al., 2018). This version introduces an interface for linking results of these speaker identification algorithms to the tool. For this, task generation in the human-assisted pipeline introduced in Section 3.3 has to be changed.

Section 5.3 describes the development process and results in more detail, while further explaining the features listed above.

4.3.4. Log Analysis and Performance Measurement

No exhaustive analysis performed in-person, such as lab or field studies, is conducted for this work. The reasons for refraining from doing such studies for efficiency analysis are explained below.

First, such studies are usually carried out in a well-controlled environment. However, as mentioned in Section 4.3.2, many users work on their own computers or even from home. Besides, work patterns of transcribers might differ when completing tasks in an unsupervised environment (bathroom breaks, distractions, browsing the web, etc.). Efficiency analysis should be as closely geared to real-world experience as possible. Log analysis is the best way to achieve that. Results of performance and efficiency analyses should be eligible values usable to derive future workload from the existing task backlog.

Transcription tasks differ vastly and depend on many factors such as video length, number of speakers, discussed bills, state, transcriber experience, and many more. Picking an "average task" to evaluate the system represents a profoundly difficult task. The same can be said about picking an "average transcriber".

Lastly, sample size can be listed as another reason why log analysis is preferable to a lab study. The huge amount of gathered logs enables analysis of an arbitrary number of transcribers and tasks. In contrast to that, Rovin (2016) conducted lab studies with only 15 transcribers which worked on an overall of five tasks (see Section 3.5.2).

To properly utilize logs, an automatic system has to be build which facilitates reading out tool and transcriber efficiency. Efficiency metrics must be formulated which enable such supervision. Performance changes should also be detectable. If feasible, project administrators receive access to a simple user interface enabling observance of these metrics.

Furthermore, analysis is conducted to identify general interaction patterns and specify how much time they take up in general (correcting text, identifying speakers, etc.). This helps classifying which component of the transcription screen causes high effort. Such knowledge is useful in consideration of future improvements. Until now, only assumptions could be made about the amount of time specific transcription processes would take.

So far, supervisors of the Digital Democracy initiative assumed the following about the manual editing process of transcripts:

- Identifying speakers and their affiliation takes up most of the transcription time
- Only minor text corrections are necessary, most of the text only has to be proofread
- Splitting up utterances is a lot more time-consuming than merging

Lastly, data such as text changes can be used to perform basic NLP analysis. While not main focus of this thesis, a proof of context that advanced research on log data is possible is carried out.

4.3.5. Release Cycle and Evaluation

Base tool rework, logging system implementation, and four separate improvement packages were defined in the sections above. The base rework is scheduled to go live in late October 2017. Afterwards, feature packages are released sequentially from November 2017 to February 2018 as four different tool versions.

Due to the necessity of publishing improvements as fast as possible to save costs, sequential A/B testing is used to evaluate changes. It is planned to collect logs for each version over a period of about two weeks. As an approximate criterion, log data of at least 500 tasks should be recorded per tool iteration. Since transcriber working times vary, exact release dates are bound to the amount of work done by transcribers. If there is not enough data to evaluate performance of a specific version, release of the next phase will be postponed by a few days. Christmas break could change the schedule further, since most transcribers are out of town and might not perform sufficient amount of work from end of December until beginning of January.

4.4. Summary

Transcription Tool's goal is to provide a user-friendly interface for carrying out and administrating the human-assisted process necessary to produce high-quality transcripts. However, current software is not ideal and can be improved. For this, changes to the code structure must be made. New features are also necessary to enable transcribers to perform work more efficiently. At last, measures offering possibilities to evaluate performance and efficiency must be taken.

To start, Transcription Tool's current backend code is refactored. For this, the obsolete Stripes framework is replaced with the Spring framework. Further code architecture changes are necessary, such as introduction of a service layer.

The Digital Democracy initiative and its supervisors lack tools to evaluate Transcription Tool performance. To solve this problem, a logging system is implemented. By using the data this system produces, metrics can be calculated. This allows software developers and administrators to monitor values such as transcription time, tool efficiency, and performance. Measuring transcription time is important to allow proper planning and estimation of the duration and cost of future transcription operations. Furthermore, such logs are useful to determine interaction patterns and how much time they consume. Besides, the huge amount of data produced might turn out to be useful for advanced research.

In addition, new features for Transcription Tool are necessary. It is decided that the following feature packages are implemented and released as four separate tool versions: speaker profile picture preview, new video player features, upgrades to utterance navigation, and VFT speaker recognition. Tool versions are released in sequential order.

Lastly, changes introduced by these new feature releases have to be evaluated. Results of the logging system are used to achieve this.

5. Implementation

Section 4.3 gave an overview of how improvements to the Transcription Tool are planned to be carried out. This chapter describes technical implementation details further. Measures taken to realize the first rework of the original Transcription Tool are defined first. Next, realization of the logging system is explained in detail. Lastly, new tool features are presented as four separate tool versions.

Implementations demonstrated in this chapter lead to completion of the first two goals defined in Section 4.1: developing Transcription Tool improvements and implementing a logging system.

5.1. Rework of Original Transcription Tool

This rework replaced the deprecated Stripes framework with Spring. Software development was performed in collaboration with the Digital Democracy project team.

5.1.1. New Tool Technology

For easier configuration and project start up, Spring Boot was introduced into the technology ecosystem (Webb et al., 2018). Over the course of this rework, data access to the MySQL database was also restructured. OrmLite was phased out and Spring Data JPA repositories as well as Hibernate were utilized instead (Gierke, Darimont, Strobl, Paluch, & Bryant, 2018; Red Hat, 2018). Corresponding deprecated code was replaced with simple JPA repository functionality. Furthermore, basic Spring AOP concepts were used to measure performance of database queries (Pivotal Software, 2018d). In addition, duplicate code and multiply defined logic in the controller layer was outsourced into an additional service layer. However, web controller code was first changed to implement Spring MVC (Pivotal Software, 2018f). Stripes' templating approach for website fragments was also replaced by Apache Tiles (The Apache Software Foundation, 2018b).

No major new features were introduced over the course of this rework. Besides the expected increase in stability and speed, only minor changes such as a dialog for person creation and editing were realized. Frontend code was not subject to revisions during this first rework. Figure 5.1 shows newly introduced tool technology components.

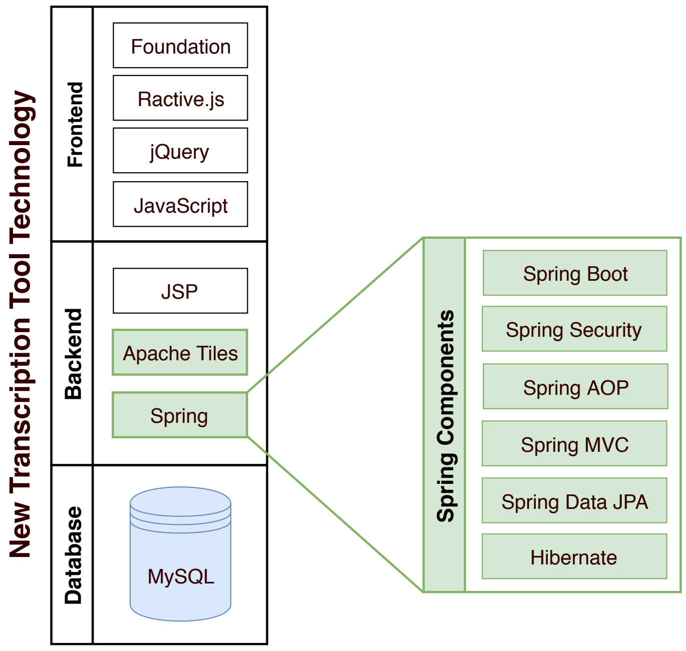


Figure 5.1.: New Transcription Tool technology. Novel components are marked in green.

5.1.2. Backend Code Architecture

Introduction of new backend components was used as an opportunity to change code architecture along the way. This aimed at improving maintainability and simplicity.

Architecture Layers

The project team agreed to try and follow the following layering approach in the code:

1. Data Access Layer
2. Service Layer
3. Controller Layer
4. View Layer

The Data Access Layer utilizes Spring JPA repositories and entities to access the underlying database. JPA entities are representations of database tables as Java classes, enabling object-relational mapping. In the Service Layer, complex business logic reused by WebControllers, RESTControllers, or other services is located. These controllers are situated in the Controller Layer, alongside exception handlers and other components handling communication between views and business logic. View templates and other components the user directly interacts with are collected in the View Layer.

Data transfer between layers is performed by either using JPA entities or Data Transfer Objects (DTOs). Since they match the database structure exactly, passing

JPA entities to views through the Controller Layer is not ideal in many cases. Services, controllers, and views might not always use the exact same fields existing in a database table. Sometimes more information is necessary, sometimes less. Therefore, DTOs were introduced to ensure better encapsulation. Data is then displayed on websites through use of JSP (WebControllers), or passed to the user's browser as JSON (RESTControllers).

Figure 5.2 visualizes code architecture layers as described above. A practical example of this architecture is given in the section below.

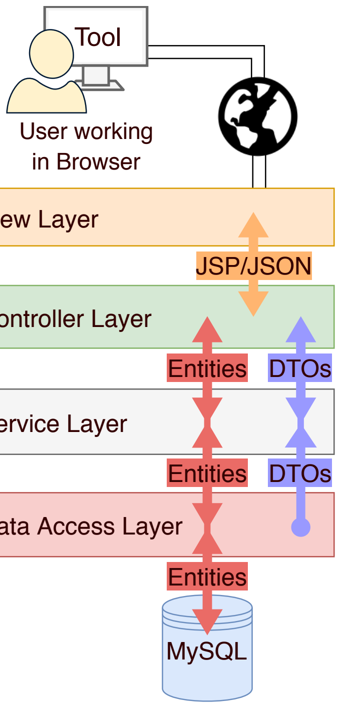


Figure 5.2.: Architecture for Transcription Tool code layering.

Layered Architecture in Practice

The following few paragraphs illustrate the process of data passing through architecture layers by using a concrete example. For this, paragraphs below describe how information about a person and their classification type is transferred from database to frontend. Exemplary code snippets were included to provide better insight and clarity, based on official Spring documentation (Pivotal Software, 2018c, 2018f, 2018a; Gierke et al., 2018; Pivotal Software, 2018e).

In the DDDB, persons of all affiliations are stored in the `Person` database table. Using MySQL, a person's classification type is defined via references to tables such as `Legislator`, `Lobbyist`, `GeneralPublic`, `LegislativeStaff`, or others. Although these tables are represented by entities in the code as well, explanations focus on `Person`.

Listing 5.1: Person entity implementing object-relational mapping.

```java
@Entity
public class Person {
  @Id
  @GeneratedValue(strategy=GenerationType.IDENTITY)
  private Integer pid;
  private String title;
  private String first;
  private String last;
  // ...
```

Listing 5.2: Excerpt of the PersonRepository interface.

```java
@Repository
public interface PersonRepository
        extends JpaRepository<Person, Integer> {
  // Default JPA Query, automatically resolved into SQL
  List<Person> findByFirstAndLast(String first, String last);

  // Custom Query using native SQL
  @Query(value = "SELECT first, last FROM Person",
      nativeQuery = true)
  List<Object[]> findOnlyNames();
  // ...
```

To achieve object-relational mapping for the `Person` table, multiple steps are necessary. First, a Java class named `Person` tagged with the Spring `@Entity` annotation has to be created. Table columns must be defined as attributes in this class. Columns representing primary keys have to be labeled with specific annotations like `@Id` and `@GeneratedValue`. Other columns can be annotated with `@Column` to modify standard JPA configuration such as default column naming. Listing 5.1 shows an excerpt of the `Person` entity class.

Next, an interface named `PersonRepository`, extending Spring's `JpaRepository`, is defined. Spring automatically initiates default JPA repository functionality on program startup for all such interfaces. This facilitates saving, querying, and deleting rows in the database without writing any further code. To enable this inherent functionality, the entity a repository handles as well as its primary key type have to be specified upon interface extension. Assuming the primary key of `Person` is `Integer`, `PersonRepository` must extend `JpaRepository<Person,Integer>`. After this, JPA repositories can perform simple queries by defining methods in written language, using specific keywords (Pivotal Software, 2018b). For example, a method named `findByFirstAndLast` automatically queries the `Person` table for the first and last name passed as String parameters. In addition, repositories can perform more sophisticated queries using the `@Query` or `@NamedNativeQuery` annotations. A code snippet of `PersonRepository` can be seen in Listing 5.2.

In most cases, DTOs only occur past the Data Access Layer. However, Spring

Listing 5.3: Code snippet extracted from PersonService.

```java
@Service
public class PersonService {
  private final PersonRepository personRepository;

  @Autowired
  public PersonService(PersonRepository personRepository) {
    this.personRepository = personRepository;
  }

  public ClassifiedPerson getClassifiedPerson(int personId) {
    // Use repository functionality to grab row by ID
    Person person = personRepository.findOne(personId);
    return findClassification(person);
  }

  // Complex functionality to find classification
  private ClassifiedPerson findClassification(Person p) {
    // ...
```

repositories allow for direct transformation of entities into DTOs upon returning rows from the database. For example, basic person data could be directly combined with classification information into a `ClassifiedPerson` object using a more sophisticated query and a custom result set mapping.

Further processing of Entities or DTOs is ideally performed in services. Annotating a Java class with `@Service` registers such a component in Spring. Defining logic in methods of these classes provides reusability and encapsulation. Although DTOs may also appear in repositories as mentioned above, transition from entities to DTOs usually happens in services. To revisit the previous example, a `Person` can have multiple classifications. Due to this reason, `@PersonService` implements a method which figures out the best-fitting classification for the current context. It then builds a `ClassifiedPerson`-DTO out of the newly found information as well as the original entity and returns it to the caller. Part of this source code is displayed in Listing 5.3.

An important aspect of Spring is dependency injection, also called "Inversion of Control" (IoC) (Pivotal Software, 2018a). Elaborating this concept in great detail would be beyond the scope of the attempted explanations. However, the reader should be familiar with its basics to understand presented code snippets.

Triggered by `@Autowired`, all object dependencies tagged with this annotation are automatically injected by the Spring IoC container. Only specific classes annotated as Spring components can utilize this functionality (`@Repository`, `@Service`, `@Controller`, etc.). Basically, such objects do not have to be instantiated manually. Instead, they are created and injected inherently by other components of the framework.

Controllers, annotated with either `@Controller` or `@RestController`, handle

Listing 5.4: Example of mappings specified in PersonController.

```
@RestController
public class PersonController {
  private PersonService personService;

  @Autowired
  public PersonController(PersonService personService) {
    this.personService = personService;
  }

  @GetMapping(value = "/person/id/{personId}")
  public ClassifiedPerson retrieveClassifiedPerson(
      @PathVariable("personId") Integer personId) {
    // Returned object is automatically converted to JSON
    return personService.getClassifiedPerson(personId);
  }
  // ...
```

communication between backend logic and frontend views. They match HTTP requests issued by the client to functionality provided by Java code. Method annotations such as `@RequestMapping`, `@GetMapping`, or `@PostMapping` realize mapping of URLs to functionality. Through these methods, generation of HTML via JSP is triggered. They can also directly return data such as JSON entities to the browser. For example, `PersonController` maps an HTTP request sent to `"person/id/42"` to a method actually calling `PersonService.findOneClassified` and returning the `ClassfiedPerson` which has an id of 42 as JSON. Listing 5.4 shows the controller code necessary to achieve this exact request mapping.

Views represent the final link between backend functionality and the frontend a user interacts with. For the Transcription Tool, this layer contains JSP and Apache Tiles functionality as well as client-server interaction using JSON. A concrete example of interaction between Controller and View Layer was already given in the paragraph above.

5.2. Logging System

As elaborated in Section 4.3.2, actions taken by transcribers have to be observable. To achieve this, interactions between transcribers and the Transcription Tool are recorded and saved into logs. This not only allows the system to keep track of the transcription progress of separate videos, but also enables performance measurements and identification of interaction patterns in a live environment. It also provides additional information for debugging.

This section starts by listing the information collected for each event. Then, actually logged events are specified.

To match the requirements defined in Section 4.3.2, up to eight values are recorded for each collected event. This collection of fields is called `LogEntry`:

- **event**

  String value describing the triggered event. This can either be a standardized JavaScript event, or a custom event defined for our purpose. Events are explained in more detail below.
- **element**

  Either the tag name of the HTML element for which the event was recorded, or additional information for custom events.
- **tagId**

  Unique identifier given to this element. This id might contain information such as the database id of specific entities to enable further analysis.
- **value**

  Text content of the targeted HTML element when the event was triggered. In case of a custom event, this field is used for providing additional information about the specific interaction.
- **keyPressed**

  Integer value describing the key pressed to trigger this event. For keyboard interactions, ASCII decimals are used. Left, middle, and right mouse buttons are represented by values 0, 1, and 2.
- **timestamp**

  UNIX timestamp in milliseconds (UTC).
- **editorId**

  Unique identifier for the current editor.
- **ip**

  IP address, used to detect if a person was working remotely or from the project lab.

To prevent the system from recording logs too excessively, only specific interactions with HTML elements and interaction patterns relevant to the transcription workflow are taken into consideration.

The following itemization lists all standard JavaScript events that are recorded. It also includes short descriptions on when they are triggered, according to Mozilla and individual contributors (2018):

- **focusin**

  An element gains focus, via mouse click or any other interaction.
- **focusout**

  An element loses focus.

- **click**

  A mouse click is performed on an element.
- **change**

  Upon losing focus, content of an element is different from when it first gained focus.
- **keydown**

  Any key is pressed down, whether they produce a character value or not. Specific Transcription Tool shortcuts are also registered and reflected in the logs using this event (for example triggering video fullscreen, jumping to next/previous utterance, etc.).

.

Events listed above are collected for the following HTML-elements on the transcription screen:

- Input fields
- Text areas
- Drop-down lists ("select"-tags)
- Links
- Buttons
- Video player and all of its interface elements
- Specific clickable markup elements (some "i" and "span"-tags)

Note that a single element can fire multiple events at once. For example, clicking on an input field which is not currently focused triggers `focusin` as well as `click` at slightly different times. After changing the content and leaving the field, `change` and `focusout` are both fired. For such standard events, collecting information is straightforward: `LogEntry` fields are filled with data grabbed directly from corresponding JavaScript event objects.

In addition to standardized browser interactions, custom events relevant for analysis are also included in the recordings:

- **load**

  Transcription screen is loaded or unloaded. Records a `LogEntry` with `value` set to either `true` or `false`, dependent on whether the screen is opened or closed.
- **visible**

  Monitors the visibility of the current window (browser tab). As long as the tab is at least partially visible in a non-minimized window, it is considered visible. If the visibility changes, a `LogEntry` is created with a `value` of either `false`, if the window was hidden, or `true`, if it became visible.

Listing 5.5: Example of a LogEntry in JSON.

```
{
  "event":       "click",
  "element":     "TEXTAREA",
  "tagId":       "textarea-20959669",
  "value":       "Mr. President, I withdraw
                 my point of order.",
  "keyPressed": 1,
  "timestamp":   1512408954042,
  "editorId":    107,
  "ip":          "123.45.6.789"
}
```

- **highlight**

  A user highlights a specific text on the website (person names, unknown words, etc.). The highlighted text is stored in the `value` field of a `LogEntry`.

- **copy, cut, and paste**

  Records copied, cut, or pasted text and stores it in the `value` field of a `LogEntry`.

- **hovering**

  Using JavaScript's `mouseenter` and `mouseleave` events, hovering specific elements is registered.

Due to the vast amount of triggered events, `LogEntries` are batched and only transmitted to the server every nine seconds. A separate file is created for each transcription task and contains JSON representations of multiple such entries. Each log file is marked with the task's unique id existent in the DDDB. A concrete example for a single JSON `LogEntry` describing an event in a log file can be seen in Listing 5.5.

Although code of the logging system was originally developed for the rework of the Transcription Tool described in Section 5.1, software code was also ported to be compatible with the old tool based on Stripes.

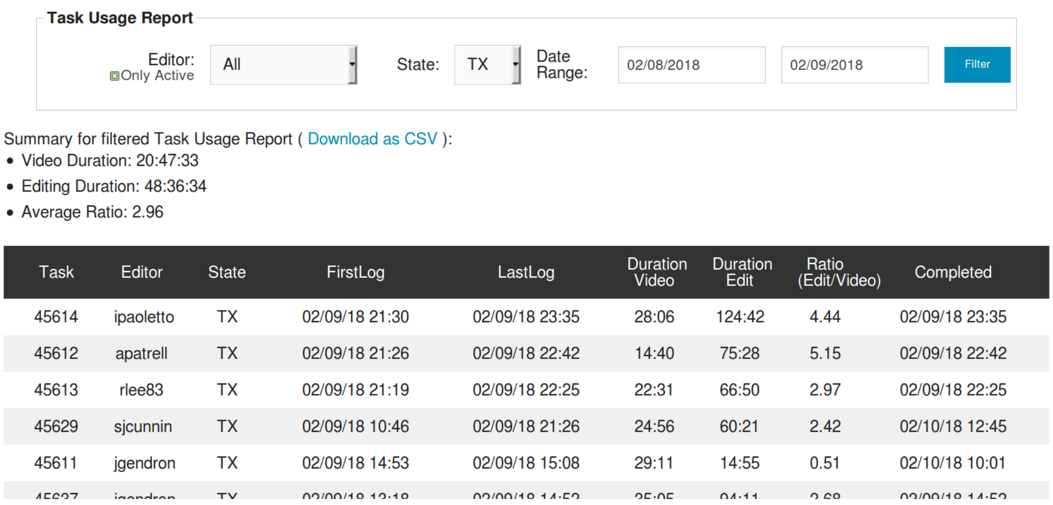
Before these logs are analyzed, they are saved to the DDDB. This enables faster processing and easier data access. Nightly executed Python scripts insert log information produced during the last day into the database. The corresponding table is named `TT_Log`. In addition to the fields existing in a `LogEntry`, the following data is also added: information about the current tool version (`tversion`), last modification date of the entry (`lastTouched`), and a hash value of the `value` field (`valueHash`). Each row also holds an auto-incremented id for indexing.

Table 5.1 displays an excerpt of entries in the `TT_Log` table. In this example, a user first clicks on a tab. Then, they modify text in an utterance text area. In the end, they leave the window. Presented rows only show columns most relevant to user interaction. Since fields such as `taskId`, `editorId`, and `tversion` would be the same for each entry in the table, they were excluded.

Table 5.1.: Example of log entries as structured in the database.

timestamp	tagId	element	event	keyPressed	value
1510367848260	quick-edit-tab	A	focusin	0	
1510367848428	quick-edit-tab	A	click	1	
1510367850622	quick-edit-tab	A	focusout	0	To protect a life of a person.
1510368051694	textarea-4452	TEXTAREA	click	1	To protect a life of a person.
1510368052358	textarea-4452	TEXTAREA	keydown	8	To protect a life of a person.
1510368052734	textarea-4452	TEXTAREA	keydown	84	To protect life of a person.
1510368052958	textarea-4452	TEXTAREA	keydown	72	To protect t life of a person.
1510368053190	textarea-4452	TEXTAREA	keydown	69	To protect th life of a person.
1510368056571	textarea-4452	TEXTAREA	change	-1	To protect the life of a person.
1510368056572	textarea-4452	TEXTAREA	focusout	0	To protect the life of a person.
1510368079199	window	window	visible	-1	false
1510368079199	window	window	load	-1	false

A basic interface was built which provides a report of primitive transcription time calculations. This usage report screen helps researchers as well as supervisors of the initiative to visualize overhead of the transcription process. Users can query for duration by editor, state, and date (via use of a date picker). The screen produces a list of results containing details about the task transcription process. This information includes: transcriber, video duration, time for transcription, ratio of transcription time to video duration, as well as time of first log, last log, and completion. Figure 5.3 shows the implemented usage report screen. Results can also be downloaded as comma-separated values (CSV) files. Values displayed in this report represent crude measurements of tool performance processed in Java. More accurate calculations using Python scripts are specified in Chapter 6 and carried out in Chapter 7.



Task Usage Report

Editor: All
Only Active
State: TX
Date Range: 02/08/2018 02/09/2018
Filter

Summary for filtered Task Usage Report (Download as CSV):
- Video Duration: 20:47:33
- Editing Duration: 48:36:34
- Average Ratio: 2.96

Task	Editor	State	FirstLog	LastLog	Duration Video	Duration Edit	Ratio (Edit/Video)	Completed
45614	ipaoletto	TX	02/09/18 21:30	02/09/18 23:35	28:06	124:42	4.44	02/09/18 23:35
45612	apatrell	TX	02/09/18 21:26	02/09/18 22:42	14:40	75:28	5.15	02/09/18 22:42
45613	rlee83	TX	02/09/18 21:19	02/09/18 22:25	22:31	66:50	2.97	02/09/18 22:25
45629	sjcunnin	TX	02/09/18 10:46	02/09/18 21:26	24:56	60:21	2.42	02/10/18 12:45
45611	jgendron	TX	02/09/18 14:53	02/09/18 15:08	29:11	14:55	0.51	02/10/18 10:01

Figure 5.3.: Transcription Tool usage report displaying results of transcription time calculations.

5.3. New Tool Features

As explained in Section 4.3.5, feature development and tool releases are performed in a sequential manner. These releases are titled "versions" for the remainder of this paper. The in Section 5.1 described revamp of the old tool is labeled "Baseline version", or version 0. It is similar to the old tool in terms of functionality.

Table 5.2 gives an overview of all released tool versions. The following subsections describe each new version and their functionality in detail.

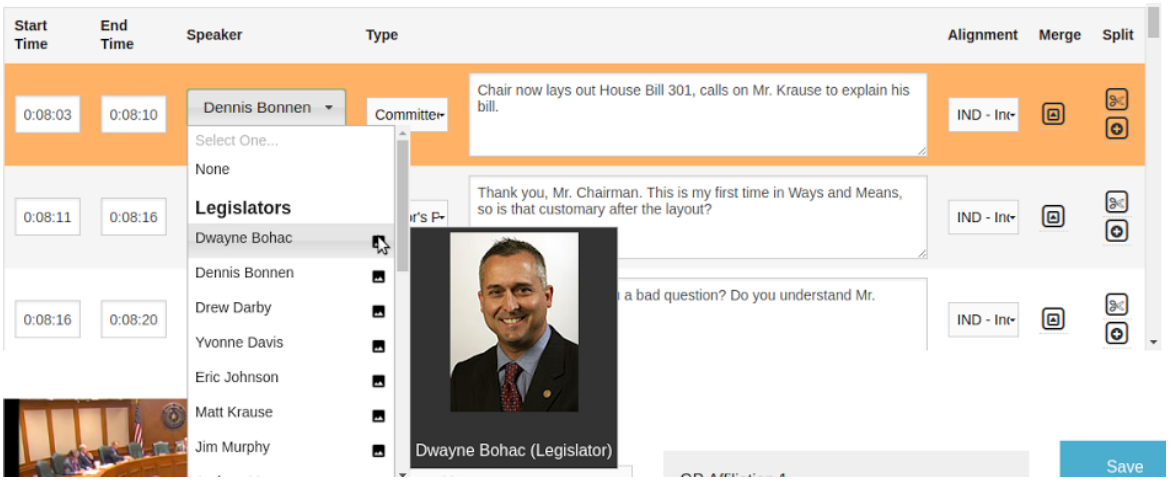
Table 5.2.: Transcription Tool versions with feature descriptions.

#	Name	Description
0	Baseline	Rework of old Transcription Tool, no major functionality changes
1	Profile Preview	Hovering icon near people's names shows preview of their profile picture
2	Video Features	Fullscreen for video player, UI to change playback rate of video
3	Utterance Navigation	Buttons for directly jumping to utterance in video and going to next or previous utterance, interactive error messages, UI for manually setting video time
4	VFT	Incorporation of voice, face, and text analysis for speaker suggestion

5.3.1. Profile Preview

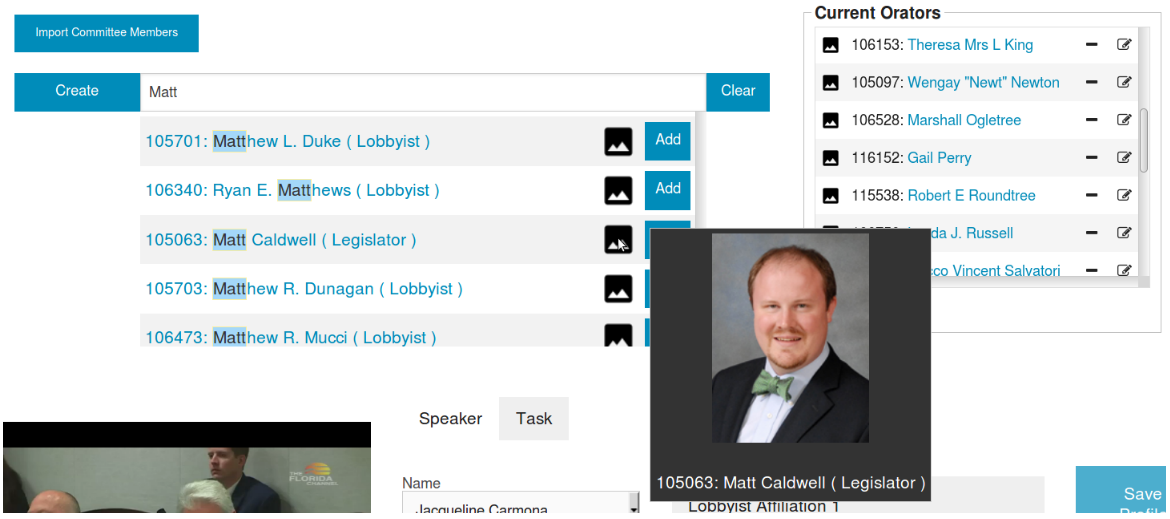
One of the biggest efforts for transcribers was named to be identifying speakers solely by their appearance on video. Especially new transcribers struggle to identify people only by picture or voice. Therefore, one of the first improvements implemented for the tool was a speaker profile preview including pictures. Through JavaScript and jQuery, this profile picture preview was made available next to speaker names in the utterance speaker selection box, speaker search result list, and orator list. For this, the speaker search area and result list was redesigned. In all mentioned UI elements, hovering the icon next to a name now displays the speaker profile picture. If no picture is available, only the person's name and classification are shown (for example legislator, lobbyist, etc.). Figures 5.4 and 5.5 demonstrate usage of the profile picture preview.

Unfortunately, for most states only pictures of legislators are freely available. These are the only images which can be retrieved effortlessly from official sites and do not put Digital Democracy at risk for copyright infringement.



Start Time	End Time	Speaker	Type		Alignment	Merge	Split
0:08:03	0:08:10	Dennis Bonnen	Committee	Chair now lays out House Bill 301, calls on Mr. Krause to explain his bill.	IND - Inc		
0:08:11	0:08:16		r's F	Thank you, Mr. Chairman. This is my first time in Ways and Means, so is that customary after the layout?	IND - Inc		
0:08:16	0:08:20			a bad question? Do you understand Mr.	IND - Inc		

Figure 5.4.: Profile picture preview for speaker selection box.



105701: Matthew L. Duke (Lobbyist) Add
106340: Ryan E. Matthews (Lobbyist) Add
105063: Matt Caldwell (Legislator)
105703: Matthew R. Dunagan (Lobbyist)
106473: Matthew R. Mucci (Lobbyist)

Current Orators
106153: Theresa Mrs L King
105097: Wengay "Newt" Newton
106528: Marshall Ogletree
116152: Gail Perry
115538: Robert E Roundtree
da J. Russell
co Vincent Salvatori

Figure 5.5.: Profile picture preview for speaker search and orator list.

5.3.2. Video Features

As described in Section 4.3.3, another frequently occurring transcriber complaint was the lack of options when interacting with the video player. This player is realized by usage of the Video.js framework (Brightcove, Inc., 2018). Speakers could sometimes not be identified due to the small size of the player. Therefore, the option to toggle fullscreen mode was implemented. This fullscreen option is describable as a “full window mode”, since one of the requirements was to allow interaction with other browser tabs while the video is maximized. Through this, currently visible speakers can be compared to people on official websites and third-party platforms. Fullscreen mode can be toggled by using the video player interface as well as a keyboard shortcut (`Alt+F`).

In addition, possibility for transcribers to change video playback rate was enabled.

Videos can be sped up and slowed down by using player interface elements and keyboard shortcuts (`Shift+"+"` and `Shift+"-"`). On the one hand, this feature aims at resolving issues with understanding mumbled notions and names. Some people speak really fast, particularly during routine interactions in House or Senate. Slowing down video playback helps transcribers to properly understand problematic audio sequences. On the other hand, some people speak very slowly, particularly during testimonies. Especially experienced transcribers can increase video playback rate to speed up their work.

Figure 5.6 shows the enhanced video player interface.

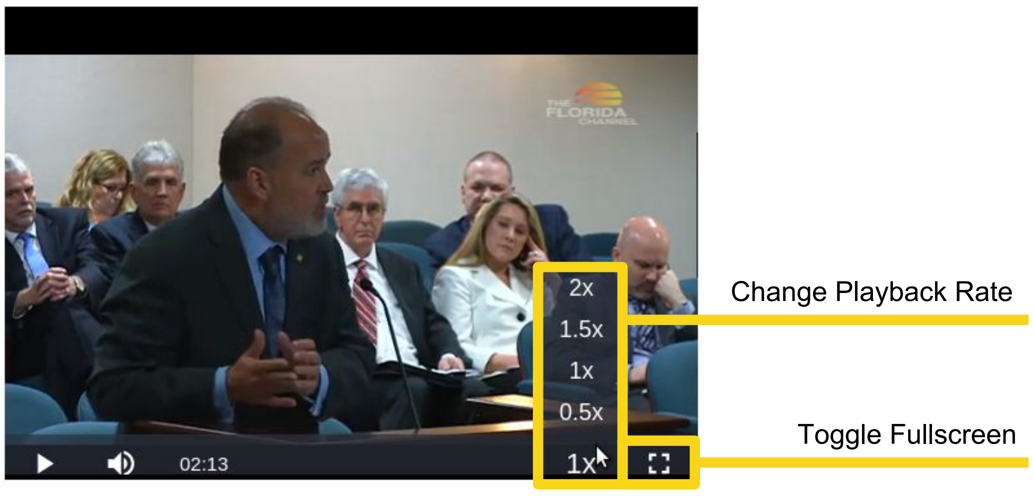


Figure 5.6.: Video player with option to change playback rate and toggle fullscreen.

5.3.3. Utterance Navigation

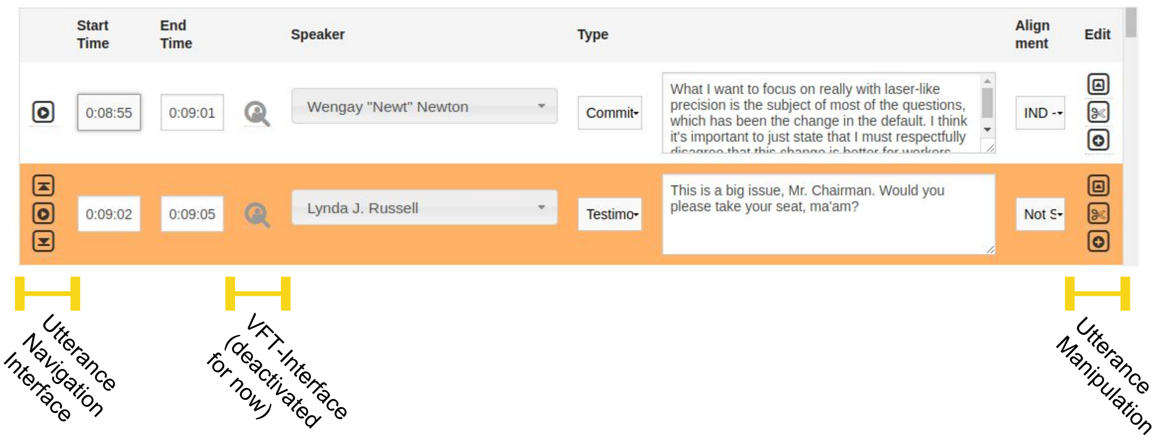
Features introduced in the previous section solve some video player problems. Improvements described in this section target the issue of transcription video navigation further. Transcribers face difficulties when trying to jump to specific utterances or sections in a video. For this, further UI improvements were developed.

Every utterance element in the transcription screen was extended with three icon buttons. The "Play"-button next to an utterance directly sets the video time to this utterance's start time. Next to the currently played utterance, two extra buttons are displayed. These allow transcribers to seamlessly jump to the previous or next utterance in the video. Keyboard shortcuts for triggering functionality of these two buttons are also available (`Alt+↑` for previous, `Alt+↓` for next utterance). Figure 5.7 shows modifications of the utterance interface element, while Table 5.3 summarizes the new icons and their functionality.

Furthermore, three input fields were appended below the video player. In these fields, a timestamp containing hours, minutes, and seconds can be entered. When pressing the "Set Time"-button, the video time is set to this exact timestamp.

Additionally, error messages displayed in the "Task"-tab of the transcription screen were made interactive. Transcribers no longer have to manually search the whole transcript for errors such as missing speaker assignments, invalid timestamps, or non-selected alignments. Clicking an interactive error message causes the utterance list to automatically scroll to the erroneous utterance. In case of missing information for speaker profiles, the interface switches to the "Speaker"-tab. These UI changes are visible in Figure 5.8.

Lastly, past tool versions introduced many new keyboard shortcuts. Therefore, a tab which contains information about all existent shortcuts was added.



	Start Time	End Time		Speaker	Type		Alignment	Edit
	0:08:55	0:09:01		Wengay "Newt" Newton	Commit-	What I want to focus on really with laser-like precision is the subject of most of the questions, which has been the change in the default. I think it's important to just state that I must respectfully	IND --	
	0:09:02	0:09:05		Lynda J. Russell	Testimo-	This is a big issue, Mr. Chairman. Would you please take your seat, ma'am?	Not S-	

Utterance Navigation Interface
VFT-Interface (deactivated for now)
Utterance Manipulation

Figure 5.7.: Updated utterance element with navigation buttons introduced in version 3.

Table 5.3.: Utterance navigation interface icons introduced in version 3.

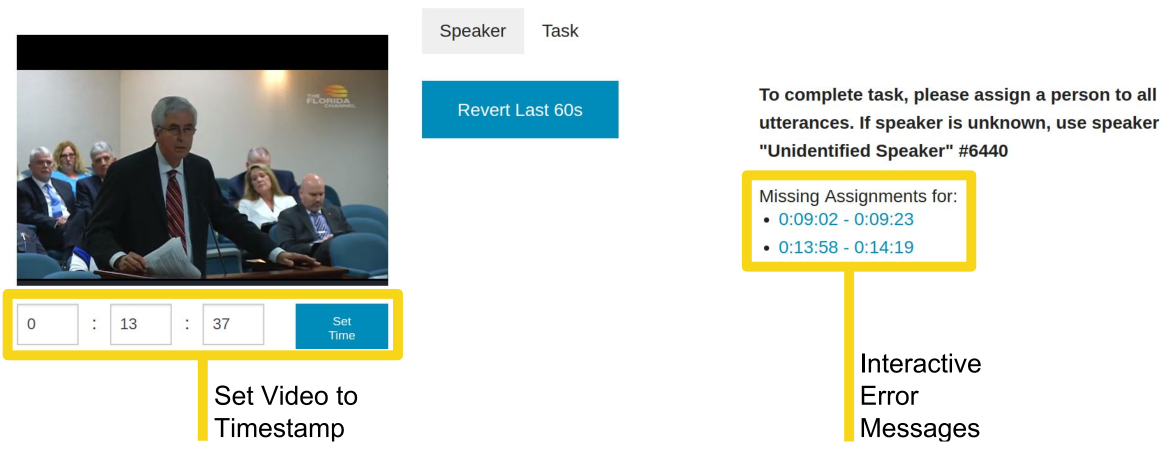
⏶	Go to time of previous utterance in video
▶	Set video time to start time of this utterance
⏷	Go to time of next utterance in video

5.3.4. Incorporation of VFT Analysis Results

As already mentioned when describing the first improvement iteration in Section 5.3.1, identifying speakers is an expensive task for transcribers. This version links the Transcription Tool to a system developed by Kauffman, Williams, et al. (2018) which implements voice, face, and text (VFT) analysis.

General VFT Functionality

The VFT algorithm is able to guess with a certain probability which speaker made an utterance. Due to the lack of data for non-legislators, speaker recognition is only



To complete task, please assign a person to all utterances. If speaker is unknown, use speaker "Unidentified Speaker" #6440

Missing Assignments for:
- 0:09:02 - 0:09:23
- 0:13:58 - 0:14:19

Figure 5.8.: Interface for setting video time and display of interactive error messages.

efficient for legislators and ex-legislators. Unfortunately, it is not effective for other speakers, such as the general public.

Speaker analysis is not performed for each utterance on its own. Most of the time, single utterances are too short for proper analysis. As mentioned in Section 3.3, speaker tags are created during preprocessing. Each of these tags is assigned to an arbitrary amount of utterances. At this point, it is not clear which person this tag represents. VFT enhances the prior diarization approach while utilizing voice, face, and text analysis to create speaker assignments for tags. It considers all utterances of a hearing which hold a specific speaker tag and tries to recognize a list of persons for this tag.

Processing Pipeline Changes

Speaker recognition using VFT is performed on a separate server, not in the Transcription Tool. In fact, speaker suggestions are not directly calculated when requested in the tool, but rather already precomputed in the preprocessing pipeline. Because of this, the human-assisted pipeline had to be changed under consideration of VFT design by Kauffman, Williams, et al. (2018). Figure 5.9 visualizes the modified preprocessing pipeline in an activity diagram. Changed components in comparison to the original design include the automatic processes of the Transcription Tool and VFT service, both on the far right of the figure.

The VFT algorithm returns a list of suggestions, each having a confidence value in percent. When the tool receives speaker suggestions during the task generation process, it saves the recommendations to the DDDB. If the VFT process or communication between servers fail for a transcription task, the system carries out a fallback to the original diarization.

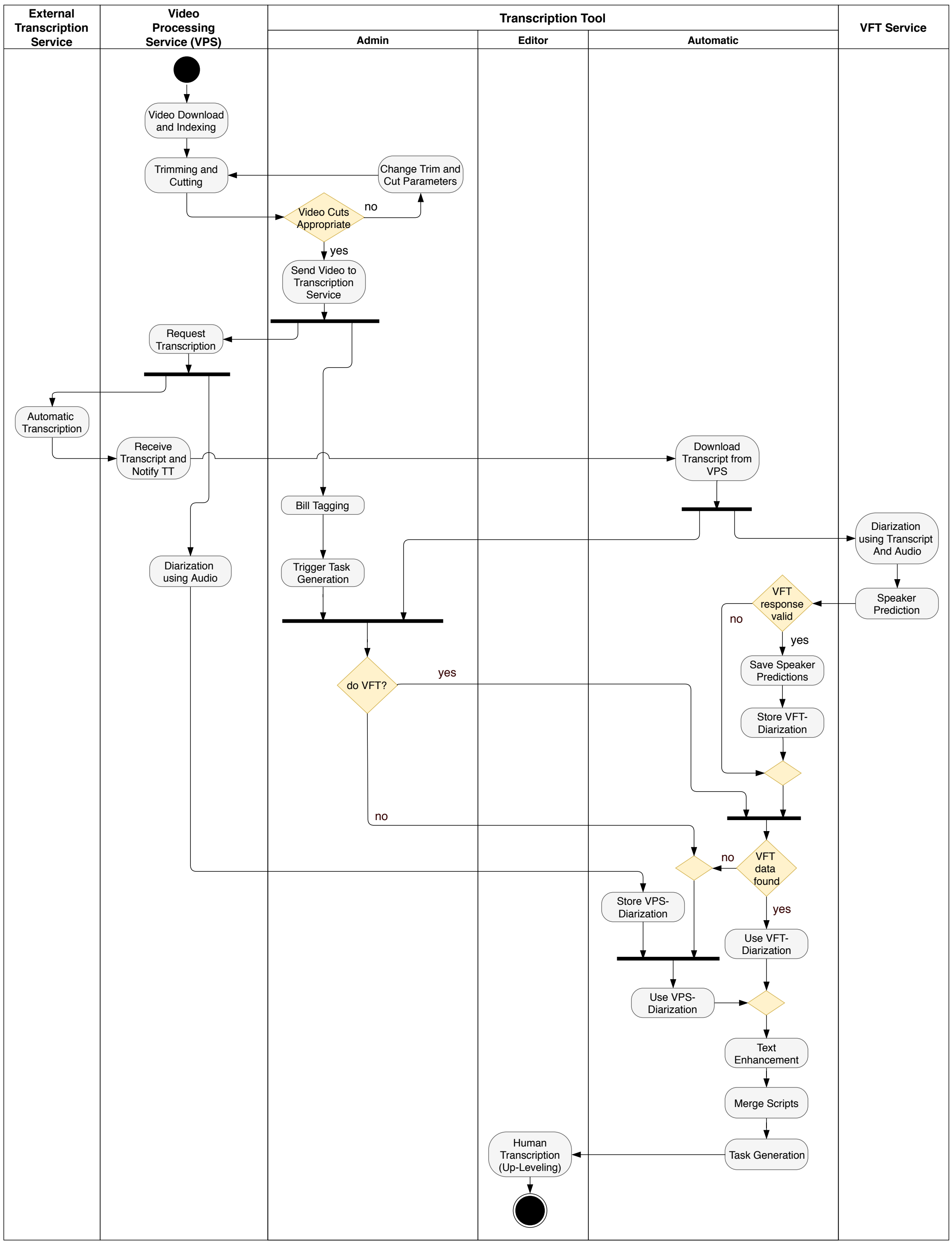
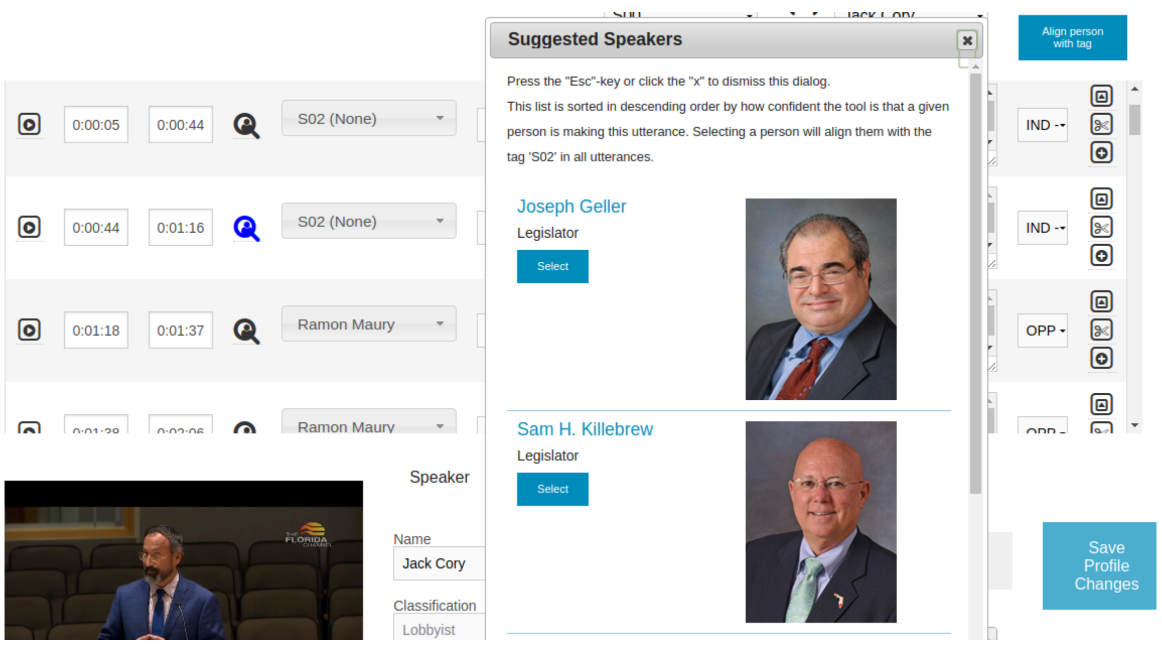


Figure 5.9.: Activity diagram of the modified Transcription Tool preprocessing pipeline. Enables VFT features and a new diarization approach by Kauffman, Williams, Washington, Socher, and Khosmood (2018).

Inclusion in Transcription Tool

To include VFT results in Transcription Tool, an icon (magnifying glass) was placed in each utterance interface element on the transcription screen. Clicking this icon brings up a dialog suggesting speakers to the transcriber.

Transcription Tool displays the five highest-confidence speaker suggestions, assuming their confidence value is greater than 5 percent. Speakers are presented in descending order, sorted by confidence. Transcribers can then choose one of the suggestions in the dialog. Selecting a speaker leads to this person being aligned with all utterances having the same diarization tag as the currently investigated one. If an utterance already holds a different speaker selection, it is not overwritten. In the dialog, speaker names are presented as links which open people's profile pages on the Digital Democracy website. Displayed profile pictures are also interactive. Clicking an image opens it in a new tab for closer investigation. If there are no speaker suggestions for an utterance, the icon triggering the dialog is grayed out. Figure 5.10 shows an example of the speaker suggestion interface for an utterance.



Suggested Speakers

Press the "Esc"-key or click the "x" to dismiss this dialog.

This list is sorted in descending order by how confident the tool is that a given person is making this utterance. Selecting a person will align them with the tag 'S02' in all utterances.

Joseph Geller
Legislator
Select

Sam H. Killebrew
Legislator
Select

Figure 5.10.: Screenshot of the VFT feature, displaying the speaker suggestion dialog. The blue icon indicates the utterance for which the suggestion is currently displayed.

Although VFT analysis is usually not providing suggestions for unknown people, mistakes can occur. This leads to the currently speaking person not always being existent in the suggestion list. Administrators can also choose to not trigger VFT analysis during task generation. If VFT is not used, no speaker suggestions are available for generated tasks.

5.4. Summary

Previously, Chapter 4 defined requirements and design of new developments for the Transcription Tool. In this Chapter, actual developments were carried out.

First, Transcription Tool had to be reworked. For this, the backend framework Stripes was replaced with Spring. Four code architecture layers were introduced: Data Access Layer, Service Layer, Controller Layer, and View Layer.

In addition to the backend rework, a logging system using JavaScript and jQuery was implemented for the frontend. It captures transcriber interactions with important elements on the transcription screen. Logs are then saved in JSON files on the transcription server. Through nightly Python scripts, they are inserted into the DDDB. A user interface was also developed which enables preview of Transcription Tool performance data.

Then, improvements to the Transcription Tool were developed and sequentially released as four new versions. First, profile picture preview was implemented by providing an icon next to speaker names. Hovering this icon shows a pop-up containing further speaker information and a profile picture. Second, video player functionality was improved. This update enabled fullscreen mode as well as slowing down and speeding up video playback rate. These functionalities were also made available to transcribers through keyboard shortcuts. Next, enhanced utterance navigation and interactive error messages were included in the tool. Utterance elements now contain three additional icons, which enable jumping to the previous, next, or any arbitrary utterance. In addition, error messages displayed in the "Task"-tab were made interactive to allow faster navigation of a transcript. Furthermore, simple UI elements were added below the video player which allow setting the video to a specific timestamp. Lastly, speaker suggestions using results of VFT analysis, based on work of previous projects and theses, were integrated. For this, preprocessing pipeline and task generation process had to be modified. Speaker suggestions are presented to transcribers in a dialog. This dialog can be triggered for each utterance by clicking a magnifying-glass icon.

6. Experimental Design

To fulfill analysis goals laid out in Chapter 4, metrics enabling evaluation of the Transcription Tool and its separate versions must be defined. This issue is addressed in this chapter by describing measurement of tool efficiency and transcriber interactions. Possibilities for detecting frequently occurring text corrections are also elaborated. Furthermore, approaches for automatic replacement of misspelled legislator names are explored. All calculations are performed based on data collected using the implemented logging system.

6.1. Transcription Tool Efficiency

To measure tool efficiency, a statistical analysis using specific metrics is performed. These metrics are coined by terms which have a distinctive meaning within the scope of this paper. They are described over the following few paragraphs.

First, it has to be mentioned that hearing video duration does not represent a reliable base value for calculation of performance metrics. In some states, videos are uploaded untrimmed and contain silent periods which do not require any transcription work. Therefore, the video speech time (VS_t) of a task (t), or duration of video containing speech, was chosen as measurement for its actual length. VS_t is derived from utterance timestamps produced by the automatic transcription process. As seen in Table 6.1, VS_t accounts for only 86.13% of the video time in a task on average. However, it differs strongly across states. Florida's videos seem to be more appropriately cut by default, since ratio of VS_t to video duration is higher (94.03%). All other states hold proportional values below 87%, with New York being the lowest at 81.6%. For all our calculations, computing values using task averages is preferable to overall mean. Longer tasks would have a bigger impact on results than shorter ones when using overall mean.

Transcription time (T_t) is the time needed to complete a specific transcription task (t), measured in minutes. There are two reasons for measuring metrics in minutes for this work. First, results of previous research, such as conducted by Rovin (2016), were also presented in minutes. However, the more practical explanation is that minutes fit the real-world context better. As explained before, transcribers work on transcription tasks affecting about 5 to 20 minutes of video. Results of calculations which aim at estimating effort or efficiency for this process are therefore better comprehensible when presented in minutes.

Although T_t can be split up into separate transcriber interactions, this section focuses on calculation of efficiency metrics. Interaction patterns are closer inves-

Table 6.1.: Comparison of video duration, video speech time (VS_t), and their relation over all states for tasks completed from October 2017 to March 2018.

State	**Tasks**	**Duration (h)**		**% of VS_t in Video**	
		Video	**Speech**	**per Task**	**Overall**
CA	697	250.26	227.79	86.86	91.02
FL	4091	683.19	645.44	94.03	94.47
NY	924	288.92	262.16	81.6	90.74
TX	6483	1071.05	887.81	82.02	82.89
Average	**3045**	**573.35**	**505.80**	**86.13**	**89.78**
All	**12195**	**2293.41**	**2023.21**	**86.3**	**88.22**

tigated in Section 6.2. Task transcription time and all metrics connected to it are calculated under the prerequisite that tasks are completed by single transcribers working with a specific tool version. Tasks which were completed using multiple tool versions or by different transcribers are excluded from the results.

Transcription ratio per task (TR_t) describes the time in minutes it takes a transcriber to work on a minute of video speech in a specific task (see Equation 6.1). Subsequently, it is the ratio of T_t to VS_t.

Transcriber editing ratio (TER_{tr}) is the average TR_t over all tasks of a specific transcriber (tr), while $N_{tasks_{tr}}$ describes the number of tasks completed by a single transcriber (see Equation 6.2). TER_{tr} can be seen as a value depicting how well a specific transcriber performs the transcription process. The smaller this value, the more efficient a transcriber is.

$$TR_t = \frac{T_t}{VS_t} \tag{6.1}$$

$$TER_{tr} = \frac{\sum_{i=1}^{N_{tasks_{tr}}} TR_i}{N_{tasks_{tr}}} \tag{6.2}$$

The main metric used to measure efficiency of the tool and its separate versions is called *Transcription cost* (TC). Lower cost naturally corresponds to higher efficiency. As seen in Equation 6.3, TC represents the average transcriber editing ratio over the number of all transcribers ($N_{transcribers}$). Basically, it is the average amount of time needed to transcribe one minute of video speech.

$$TC = \frac{\sum_{i=1}^{N_{tr}} TER_i}{N_{transcribers}} \tag{6.3}$$

All metrics as well as the variables displayed in Equations 6.1, 6.2, and 6.3 can be calculated over an arbitrary subset of transcription tasks and editors (for example only for specific tool versions). When diminishing task selection according to a specific criteria (c), resulting transcription cost is referred to as TC_c.

6.2. Transcriber Interactions

The transcription process can be split up into separate interactions. Therefore, T_t is the time a transcriber needs to perform these interactions. It includes startup time, text correction, speaker identification, splitting and merging utterances, as well as passive interactions.

Equations 6.4 and 6.5 show a summarization of how T_t was defined for this analysis. Its separate components are closer elaborated below.

$$\begin{aligned} T_t = Startup_t + TextCorrection_t + SpeakerId_t + \\ Split_t + Merge_t + Passive_t \end{aligned} \tag{6.4}$$

$$Passive_t = Proofread_t + Idle_t \tag{6.5}$$

Startup time can be classified as the time span between loading the transcription screen and first user interaction.

Text correction time is probably the most elementary interaction between transcriber and tool. It records time spent in text areas editing utterance texts.

Time needed for speaker identification includes all activities belonging to speaker selection, speaker search, and correction of wrongly assigned speakers. More complex interactions have to also be accounted for, such as leaving the transcription screen to look up people on other websites.

Splitting and merging utterances are relatively simple operations. However, counting time necessary to complete them requires some consideration. Clicking the split or merge button barely takes any time, but deciding whether or not to do so does. Usually, it is obvious to transcribers which very short utterances should be merged right away. On the other hand, splitting up utterances requires more effort. Transcribers must not separate a speaker's line of thought while making sure utterance texts are of appropriate length. Due to these reasons, both operations are given fixed costs. Based on real-world observations, it was decided that merging accounts for half a second of transcription time, while splitting accounts for one second.

Besides the distinctive interactions explained up until now, others can not be measured reliably via log data. Such activities include the transcriber proofreading text and simply watching the video, or not doing any work at all (being idle). These passive activities were combined into a single parameter called $Passive_t$, as seen in Equation 6.5.

Values making up transcriber interaction time in a task can also be collected in a vector I_t (see Equation 6.6). Consequently, I_{t_i} refers to the time spent for a specific interaction type (i). Using this representation, Equation 6.4 can be shortened to Equation 6.7, which shows another definition for T_t.

$$I_t = \begin{pmatrix} Startup_t \\ TextCorrection_t \\ SpeakerId_t \\ Split_t \\ Merge_t \\ Passive_t \end{pmatrix} \quad (6.6) \qquad T_t = I_{t_1} + I_{t_2} + I_{t_3} + I_{t_4} + I_{t_5} + I_{t_6} \quad (6.7)$$

Thus far, only time spent on interactions in single tasks was considered. However, it is more relevant to find out how each interaction affects the time necessary to complete all transcription tasks. In addition, changes over different tool versions must be calculated to evaluate new features.

For each tool version, transcription time is split up into separate interactions. Each of these interactions is represented by a ratio. Changes in interaction patterns are made detectable this way. These average interaction ratios (AIR_{v_i}) are computed per tool version (v) and interaction. As seen in Equation 6.9, averaging interaction ratios (IR_{t_i}) of all tasks in a version leads to this value. To be more precise, IR_{t_i} stands for the duration of time spent for given interactions (I_{t_i}) in relation to video speech time of a task. Equation 6.8 visualizes these variables. It can be stated that AIR_{v_i} is a general representation of how time consuming an interaction type was in a specific version.

$$IR_{t_i} = \frac{I_{t_i}}{VS_t} \quad (6.8) \qquad AIR_{v_i} = \frac{\sum_{t=1}^{N_{tasks_v}} IR_{t_i}}{N_{tr}} \quad (6.9)$$

To quantify how different interactions attribute to concrete changes in timing, a parameter α_{v_i} was added for each interaction feature. α_{v_i} acts as a multiplier denoting the increase or decrease in time used for an interaction produced in this version compared to the base version 0. AIR_{v_i} is used to calculate this value. Values less than 1 represent an increase in efficiency, or reduction of cost, compared to version 0. Equation 6.10 shows how α_{v_i} is computed.

$$\alpha_{v_i} = \frac{AIR_{v_i}}{AIR_{0_i}} \quad (6.10)$$

Consecutive performance changes over multiple versions illustrated by α_{v_i} can be best displayed in a matrix. Each cell contains a concrete value for α_{v_i}, with rows representing different tool versions and columns depicting different interactions. Interaction columns are labeled in the same order as in I_t. These matrices are named interaction transformation matrices, or IT_c, with c representing an arbitrary criteria. This criteria describes how tasks for metric calculations were selected, for example by transcriber cohort or specific states. Equation 6.11 shows the layout of such a matrix, for m versions and n interaction types. As already defined above, calculations for α_{v_i} are carried out for five different versions and six interaction types. It therefore produces 5x6 matrices.

$$IT_c = \begin{array}{cl} \xrightarrow{\text{interaction types}} & \\ \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \alpha_{1_1} & \alpha_{1_2} & \cdots & \alpha_{1_n} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{m_1} & \alpha_{m_2} & \cdots & \alpha_{m_n} \end{pmatrix} & \Bigg\downarrow \text{versions} \end{array} \tag{6.11}$$

Lastly, another alternative of T_t interpretation is presented. α_{v_i} is introduced as a task-independent variable which is used to measure interaction changes over different versions. However, it can also be utilized to model I_{t_i} with the help of a variable describing the base effort needed for specific interactions, regardless of version (β_{t_i}). This variable can not be fully specified, since it is dependent on concrete task characteristics such as transcriber, video time, number of speakers, and others. Still, it can be used to model T_t for arbitrary tool versions and tasks. This statistically informed model for T_t is shown in Equation 6.12.

$$T_{t_v} = \alpha_{v_1} \cdot \beta_{t_1} + \alpha_{v_2} \cdot \beta_{t_2} + \alpha_{v_3} \cdot \beta_{t_3} + \alpha_{v_4} \cdot \beta_{t_4} + \alpha_{v_5} \cdot \beta_{t_5} + \alpha_{v_6} \cdot \beta_{t_6} \tag{6.12}$$

6.3. Transcription Text Correction

Collected log data is useful for measuring metrics such as transcription cost or transcriber interaction patterns. However, there are other conclusions which can be drawn from these massive amounts of data.

Changes to transcription texts are documented in detail every time a user presses a key. Using this data, basic research for detecting common replacements can be conducted. If frequent corrections are found, they can be automatically carried out during preprocessing. Furthermore, analyzing logs for this purpose serves as proof of concept for more sophisticated research. This was a goal laid out in Section 4.1.

6.3.1. Common Corrections

As a first step, common replacements transcribers make must be determined. This is necessary because no such research was previously performed for the Transcription Tool.

Parts of the Python library `difflib` are used to assist with this issue (Python Software Foundation, 2018). This library implements an improved version of the gestalt pattern matching algorithm introduced by Ratcliff and Metzener (1988). It calculates differences in text similar to how a version control system such as GitHub does (GitHub, Inc., 2018). Listing 6.1 shows example output for processing

Listing 6.1: Output for an example sentence comparison using difflib.

```
--- We're united and form opposition against the Legislator.
+++ We're united in firm opposition against the Legislature.
@@ -4,2 +4,2 @@
-and
-form
+in
+firm
@@ -9 +9 @@
-Legislator
+Legislature
```

differences in two sentences using `difflib.unified_diff`. This output must be postprocessed to render it useful for the previously described purpose.

6.3.2. Legislator Name Correction

Results of the general text replacement analysis will lead to concise findings about commonly corrected words. Aside from that, misspelled names being produced by text-to-speech algorithms is a known issue for Digital Democracy. However, there are no measurements yet for how often names are actually misspelled. Therefore, it is not obvious if implementing sophisticated name correction mechanisms would be cost efficient. Analysis has to be carried out that aims at identifying a need for such improvements.

Although it is possible, deciding if a word actually represents a name in a certain context is a challenging issue. Therefore, only legislator last names highly probable to appear in the current task should be considered. This can be achieved by utilizing a task's hearing data in the DDDB. Hearings are held by specific committees. Certain legislators are assigned to committees on a permanent basis during a session year. By using this information, names of committee members can be investigated for this initial research.

Dictionary Replacement

One of the easiest ways to correct names is a simple rule-based approach. A dictionary is built by collecting common correction findings. If a name is frequently modified in a specific way, adjustments can be performed automatically during preprocessing.

A dictionary is constructed using words which were replaced by committee member names. However, not every entry in this dictionary can be seen as relevant. Due to this, a threshold describing how often corrections for a committee member must occur to be considered valid is defined (θ_{dic}). For now, it is assumed that a correction has to at least happen twice for a name in a specific committee, leading to $\theta_{dic} = 2$.

Automatic Name Correction

A dictionary of common name replacements is a reliable correction tool. However, arguments can be made for attempting more advanced approaches.

When looking at transcriptions completed in the past, it becomes obvious that small mistakes in legislator last names were not always corrected. For example, automatic transcription sometimes mistakes legislator “Hertzberg” as “Hetzberg”. Transcribers seem to overlook such small typos at times. Therefore, it would not come up in a dictionary of common replacements. Legislators also change on a regular basis. Each session year could bring a different list of legislators. This would lead to problems for tasks completed in the beginning of a session year, since the replacement dictionary would not be up to date yet. Subsequently, automatic detection of misspelled names has to be performed.

As the simplest approach, all capitalized words coming out of the preprocessing pipeline can be considered possible candidates for legislator names. However, this might prove to be too general.

Because of this, more progressive techniques enabling detection of names using POS- and NER-tagging are established. Stanford taggers as well as the spaCy framework provide functionality to carry out such tagging. These technologies were already introduced in Section 2.3. After tagging sentences, two types of words are subject to investigation: Proper nouns (“NNP”-tag) for POS, persons (“PERSON”-tag) for NER.

For Stanford taggers, the `english-left3words-distsim`[1] model (POS) and `english.all.3class.distsim.crf.ser`[2] classifier (NER) are used. Processing with spaCy is carried out under support of the `en_core_web_sm`[3] model.

After candidate words (w_c) are detected using such taggers, an attempt has to be made to match them to actual legislators being part of the current committee (w_l). For this, Levenshtein and Jaro-Winkler algorithms are used.

For Levenshtein, a combination of distance and ratio (L_{ratio}, see Equation 2.1) is utilized. The candidate word which can be transformed into a legislator name with the smallest distance value should be chosen. Substitution cost of letters is set to 2 in the Levenshtein calculation, while overall distance between w_c and w_l should not exceed 3. A threshold θ_{Lr} is also introduced for Levenshtein ratio. If the chosen transformation has $L_{ratio_{w_c,w_l}} \leq \theta_{Lr}$, it is considered a match. Although an optimal value has to be found via concrete test runs, anywhere between 0.23 and 0.27 seems appropriate as an initial guess for θ_{Lr}. Using a ratio as well as a substitution cost of 2 aims at preventing premature replacement of short words.

For Jaro-Winkler, thresholds (θ_{JW}) between 0.86 and 0.90 are used as a starting point for investigation. Furthermore, distance of w_c and w_l must be less than 3 to be considered a valid replacement.

[1]https://github.com/stanfordnlp/CoreNLP/blob/master/doc/ner/README.txt
[2]https://github.com/stanfordnlp/CoreNLP/blob/master/doc/ner/README.txt
[3]https://spacy.io/models/en#en_core_web_sm

Evaluation

Results of automatic correction approaches utilizing dictionary replacement, capitalization, POS-tagging, and NER-tagging as described in the previous sections must be evaluated. For this, prerequisites have to be set first.

For each state, five tasks are selected as a test set. Chosen tasks should concern hearings of different committees to prevent bias towards specific legislator names. In addition, tasks should be randomly picked with respect to mean and median of utterance count and video speech duration of all tasks. The average and median values for video speech duration are 9.9 and 4.7 minutes. For utterance count, calculations for mean and median yield 47 and 29. Due to these values, tasks containing between 30 and 50 utterances and 5 to 10 minutes of video speech are randomly selected.

In the context of this evaluation, each transcription text goes through four iterations: unedited (u), automatically corrected (c), transcribed (t), and proofread (p). Unedited texts are directly pulled from the database after they were emitted by the preprocessing pipeline. Automatically corrected texts are those subjected to replacement approaches described in this section. Transcribed texts were processed by transcribers and marked as completed in the database. The proofread phase was introduced only for this evaluation. All transcribed tasks selected to carry out this experiment must be manually checked for errors in legislator names. This manual proofreading step is necessary because an initial investigation revealed that transcribers seem to frequently miss misspelled names. Without having access to an immaculate test set, an evaluation is not feasible.

The number of legislator last name differences in texts is represented by $\Delta_{i,j}$, where i and j mark which text iterations are compared. For example, $\Delta_{c,p}$ stands for the difference between automatically corrected and proofread version of the text.

Multiple factors must be considered when determining effectiveness of attempted corrections. Values for $\Delta_{t,p}$ have to be manually collected during proofreading and show the amount of misspelled names missed by transcribers.

$\Delta_{u,c}$ measures the amount of automatically replaced names and provides information about correction quantity, including true and false positives.

Erroneous replacements (false positives) are detectable by applying the current automatic correction approach to proofread text. Then, differences are compared to those producing $\Delta_{u,c}$ to determine the exact error count (ϵ).

Valid name corrections missed by automatic replacement algorithms (m), or false negatives, can be captured by investigating $\Delta_{u,c}$ and ϵ.

Ultimately, evaluation must focus on maximization of transcription quality and therefore keeping ϵ and m low, while attempting to produce a sufficient number of replacements ($\Delta_{u,c}$). For this, error ratios $E = \epsilon / \Delta_{u,c}$ can also be utilized as a comparison tool.

6.4. Summary

When performing a tool evaluation, clear metrics are necessary. The main goal of the attempted evaluation is to measure differences in developed tool versions using log data.

To achieve this, efficiency metrics are defined. The main metric for measuring tool efficiency is TC, or transcription cost. A decrease in cost equals an increase in efficiency. TC represents the average amount of time in minutes a transcriber takes to transcribe one minute of video speech.

Furthermore, transcriber interactions are investigated. Transcription time is split up into six different interaction types: startup time, text correction, speaker identification, splitting and merging utterances, as well as passive interactions. Metrics which enable tracking of changes in interaction patterns over different tool versions are also described.

Finally, approaches for performing automatic text correction in transcripts are introduced. Text corrections are analyzed and a common replacement dictionary is built. In addition, mistakes in legislator last names are found by utilizing capitalization or POS- and NER-tagging. Misspelled names are matched to legislators using Levenshtein or Jaro-Winkler distance algorithms. Erroneous names can then be replaced by names determined to be valid corrections. To evaluate effectiveness of replacements, proofread transcription texts must be created and compared to automatically corrected texts.

7. Findings and Discussion

In this chapter, log analysis results are presented. Collected logs were analyzed to determine results for metrics defined in Chapter 6. All computations were carried out using Python3, while accessing data stored in the DDDB.

Transcriber working time was inspected by converting log timestamps into observable data. From this, transcription tool efficiency and transcriber interaction patterns were derived. Results of these computations are discussed and investigated below. Besides this analysis of transcription time, possibilities for automatic text correction are presented.

Findings of this chapter complete the last two goals defined in Section 4.1: evaluating Transcription Tool improvements and proving usability of logs for future research.

7.1. Transcription Time Analysis

As seen in Table 6.1, 12,195 transcription tasks were completed between October 2017 and March 2018. This accounts for logs about over 2,293 hours of hearings. Since the old Transcription Tool technology was still in use through the beginning of October, only 10,464 tasks were completed with tool versions 0 to 4.

A cohort of 20 transcribers was selected to compute tool efficiency metrics and transcriber interactions. These students worked with all tool iterations, including the old tool prior to the rework. This selection aims at reducing effects of external factors such as transcriber experience. Working with tool versions 0 to 4, this transcriber cohort completed 3,374 tasks. Furthermore, tasks with $1 < TR_t < 12$ were excluded from the analysis to reduce effects of outliers. After this, 3,272 tasks were left, equaling 652 hours of hearing video and 1,605 hours of transcription work.

T_t was calculated by analyzing timestamps of all entries in the `TT_Log` table triggered by a `focusin`, `click`, `focusout`, `load`, or `visible` event. Whenever possible, transcriber inactivity was detected by `load` as well as `visible` events and excluded from T_t. To achieve more accurate results, another way of recognizing inactive transcribers was introduced. If no tool interaction occurred for more than 7 minutes, it was assumed that transcribers stopped working on the task. This concrete inactivity threshold was defined after discussions with project administrators who know the transcription process well.

7.1.1. Transcription Tool Efficiency

Table 7.1 shows a cost reduction of 16.74% from version 0 to version 2 for the selected transcriber cohort. Although still producing better performance values than version 0, version 3 caused a cost increase of 7.35% in comparison to version 2. Version 4 again leads to decreasing costs and therefore an overall increase of efficiency.

Table 7.1.: Transcription cost for the selected transcriber cohort.

Version	*TC*	*TC* Change	
		To Version 0	To Previous
0	3.935	-	-
1	3.626	7.85%	7.85%
2	3.276	16.74%	8.88%
3	3.566	9.38%	-7.35%
4	3.172	19.40%	10.01%

Figure 7.1 visualizes results presented in Table 7.1 by comparing *TC* for each version and all transcribers in the cohort. Performance changes over different versions in relation to version 0 can be seen in Figure 7.2.

Not every state which Digital Democracy processes had hearings eligible for transcription when new versions were released. Therefore, a separate analysis was performed to compare tool efficiency per state. Table 7.2 presents an overview for *TC* per state and version, while Figure 7.3 provides a visualization of these costs. Looking at this data, New York and California hearings take considerably longer to transcribe in version 3 than they do otherwise. While values for Florida barely change over different versions, Texas results show a steady decrease in cost.

Table 7.2.: Transcription cost per state and version for the selected transcriber cohort.

Version	State Results			
	California	Florida	New York	Texas
0	2.410	-	2.438	3.925
1	2.478	3.147	-	3.514
2	-	3.336	2.042	3.447
3	3.176	3.134	3.756	3.292
4	2.645	3.045	3.060	2.933

7.1.2. Transcriber Interactions

Figure 7.4 shows the amount of transcription time each interaction type accounts for, based on logs generated for version 0. These initial findings confirm assumptions of administrators about the transcription process mentioned in Section 4.3.4.

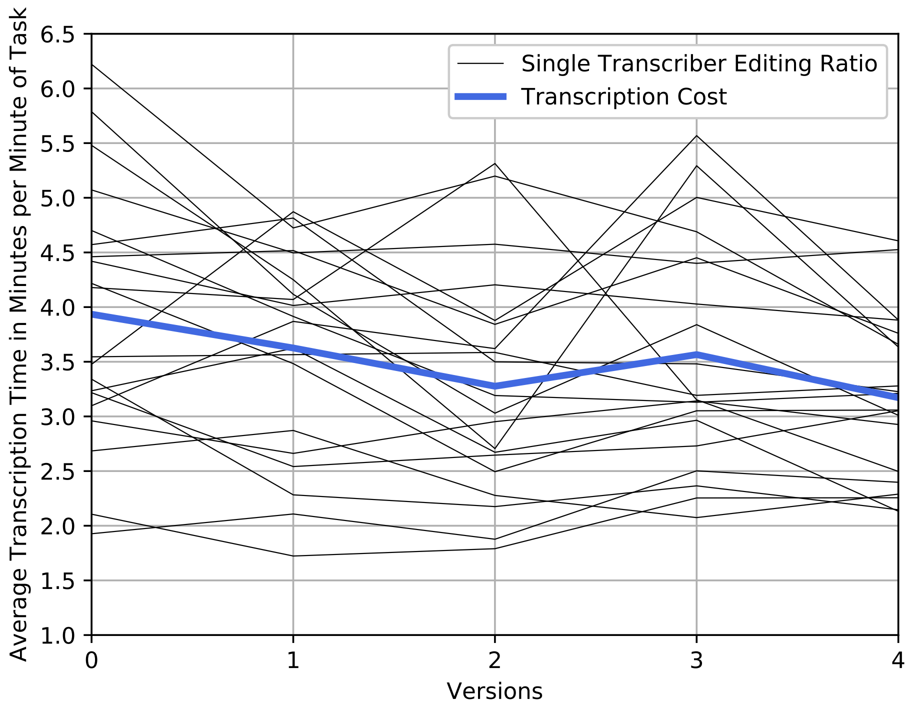


Figure 7.1.: Transcription cost per version for the selected transcriber cohort.

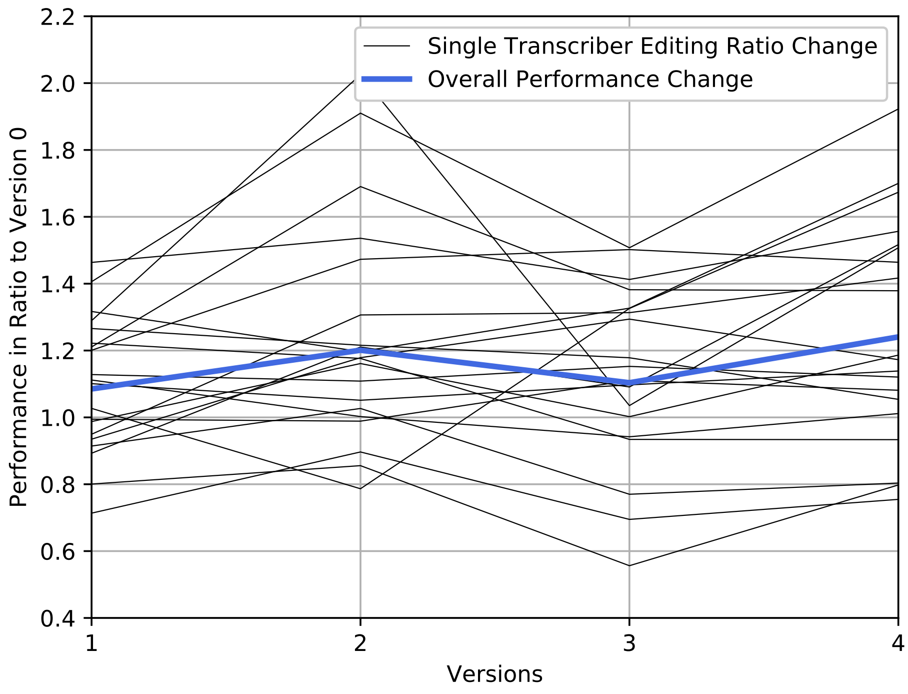


Figure 7.2.: Cohort performance change per version in comparison to version 0.

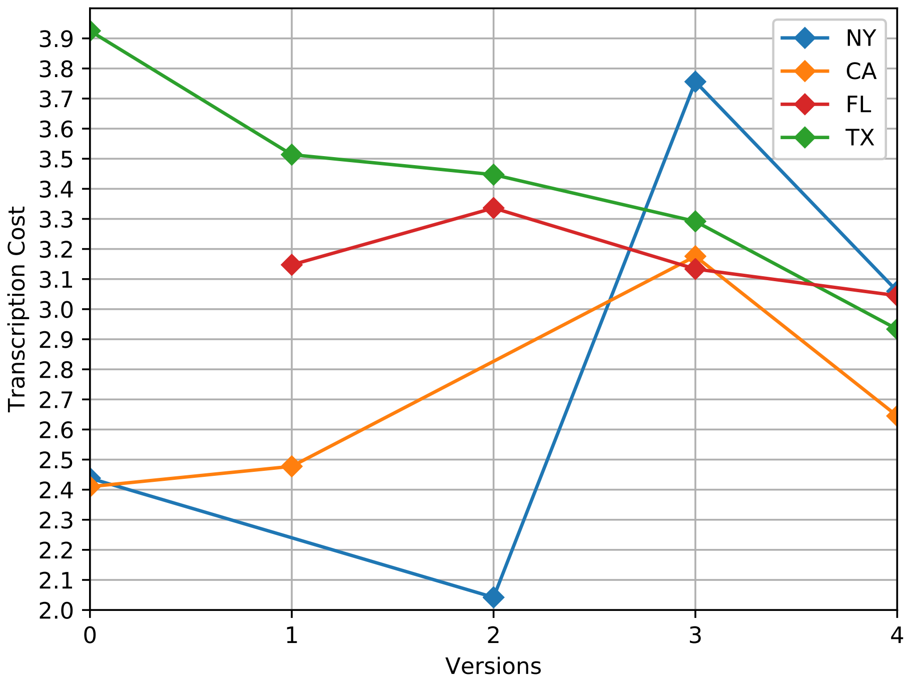


Figure 7.3.: Transcription cost per state and version for the selected transcriber cohort.

As suspected, performing text corrections represented an inexpensive task, taking up only 15.2% of transcription time. On the contrary, speaker identification consumed 34.6% of a transcriber's work time and therefore poses a problematic issue. The amount of time passing until a transcriber first interacts with the Transcription Tool was surprisingly high. Ratio of this startup time was observed to be 9.5%. Since timestamps have to be adjusted manually, splitting up utterances is considered more time-consuming than merging (4.3% versus 0.7%). However, measuring these two operations proved difficult and the approximation described in Section 6.2 was used. Passive interactions such as proofreading, watching the video, and inactivity accounted for 35.6%. The inactivity threshold was set to 7 minutes, same as for the efficiency analysis described in Section 7.1.1 above.

Figure 7.5 displays interaction analysis results over all versions. It summarizes how much each interaction type contributed to overall transcription time of all tasks. For this, results for each interaction type are clustered together. Figure 7.6 shows the same data, but aggregates interaction ratios by version instead. Both graphs enable observance of variations in interaction patterns over multiple versions.

One of the striking changes observable in the graphs is the shift in split and merge operations after version 2. Over the first three versions, merging utterances took up 0.83% of time on average, while splitting accounted for about 3.8%. For versions 3 and 4, merging heavily increased to 3% on average, whereas splitting decreased to 1.7%. Number of text corrections also declined, reaching a 11.5% low in version 4. Being especially high in version 0 with 9.5%, startup time steadily

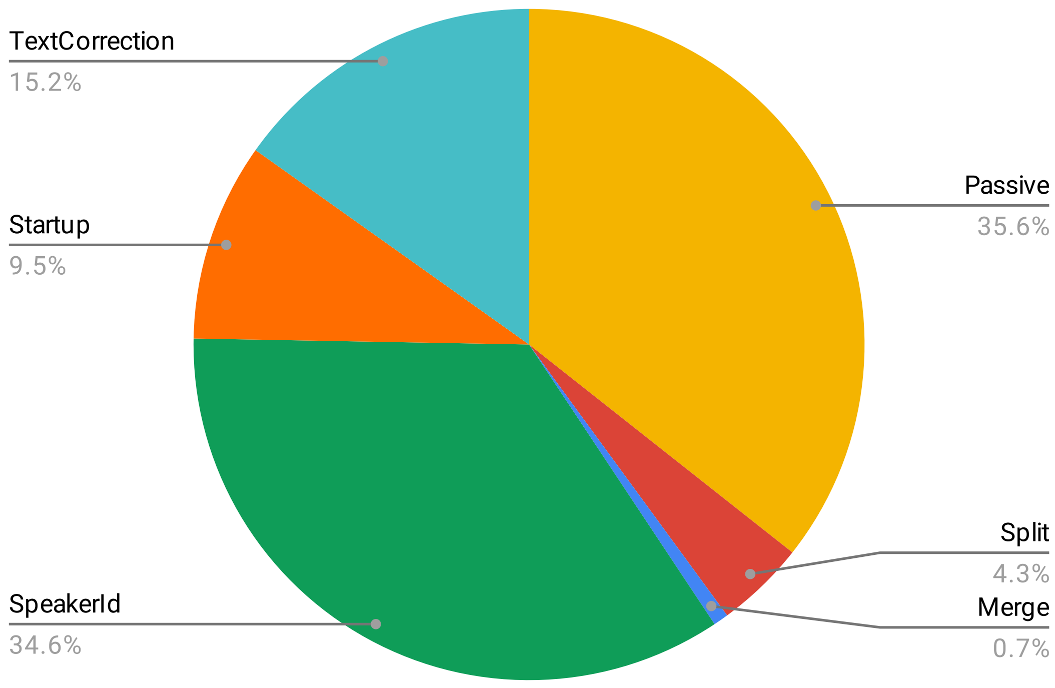


Figure 7.4.: Average interaction ratios for transcription time in version 0.

decreased to 6.2% in version 4. Speaker identification fluctuated between 35% and 28% over all versions. Versions 0 and 1 produced values around 34%, before version 2 reduced the speaker identification ratio to 28.8%. Version 3 increased it to 33.4% again, before it fell to 29.8% in version 4.

In Equation 7.1, IT_{Cohort} creates a representation of the described cohort results across all versions. On the other hand, Equation 7.2 displays a matrix for results exclusively produced by tasks handling Texas hearings ($IT_{CohortTexas}$). As explained in Section 6.2, rows in IT matrices stand for different versions, while columns represent interaction types. From left to right, these interactions are: startup, text correction, speaker identification, splitting, merging, and passive.

$$IT_{Cohort} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0.777 & 0.958 & 0.983 & 1.122 & 0.954 & 1.097 \\ 0.765 & 0.966 & 0.831 & 1.297 & 0.683 & 1.274 \\ 0.730 & 0.883 & 0.966 & 3.984 & 0.481 & 1.157 \\ 0.648 & 0.755 & 0.861 & 4.343 & 0.327 & 1.347 \end{pmatrix} \tag{7.1}$$

Because only 7 tasks concerning Texas legislature were completed with version 4, values in the last row of $IT_{CohortTexas}$ are inconclusive. In contrast to IT_{Cohort}, $IT_{CohortTexas}$ shows a constant decrease for speaker identification time in the third column. Furthermore, Texas tasks completed with version 3 had significantly higher splitting and lower merging ratios than tasks of other states. Startup time was also lower for version 3 in $IT_{CohortTexas}$.

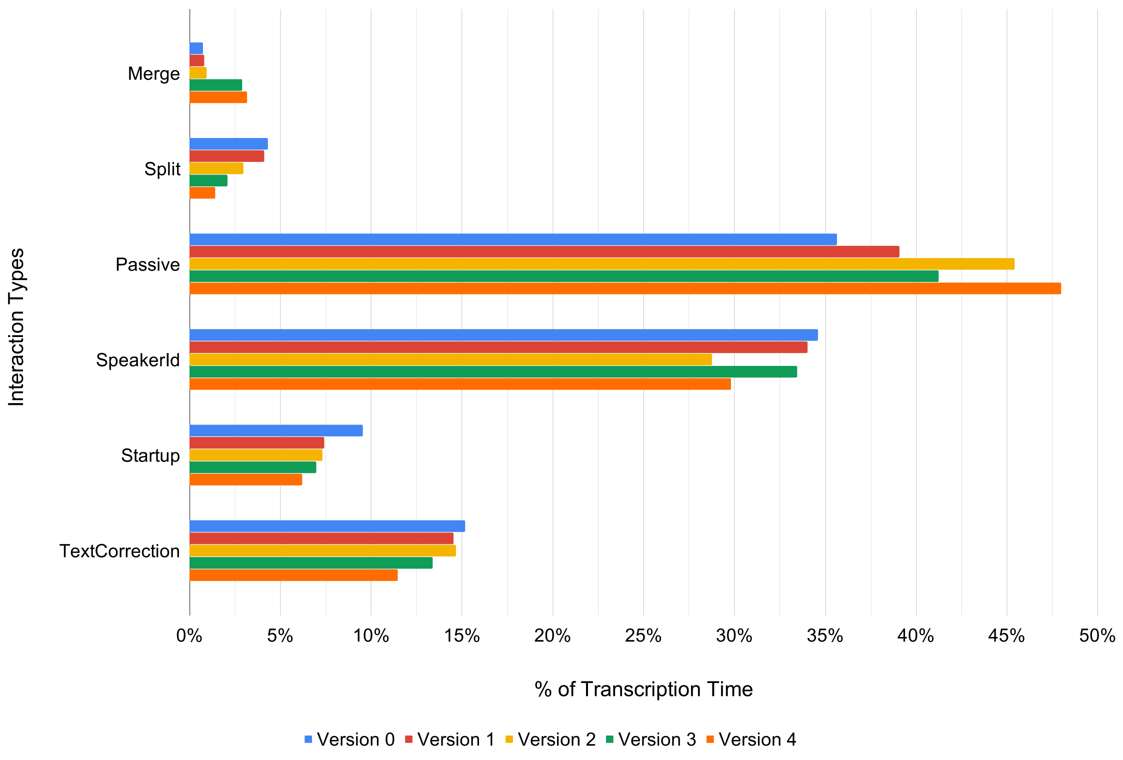


Figure 7.5.: Average interaction ratios by interaction type.

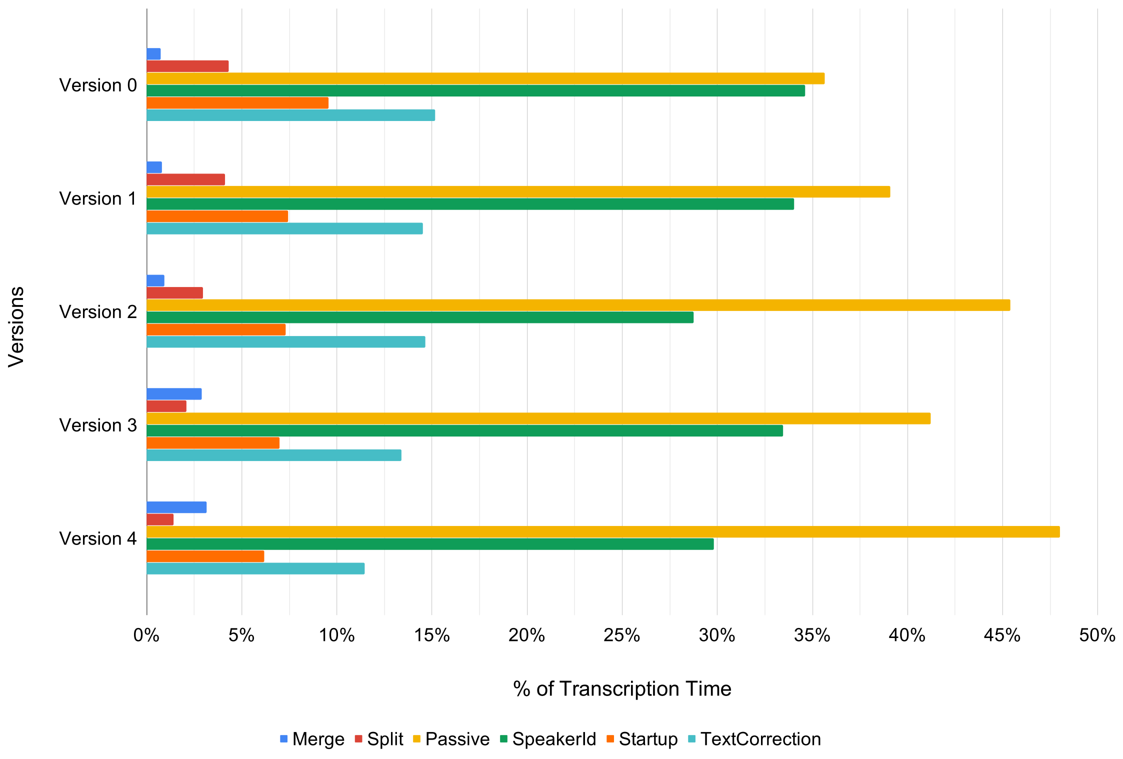


Figure 7.6.: Average interaction ratios by tool version.

$$IT_{CohortTexas} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0.891 & 0.931 & 0.926 & 1.042 & 0.857 & 1.177 \\ 0.981 & 0.861 & 0.923 & 1.301 & 0.715 & 1.202 \\ 0.481 & 1.124 & 0.747 & 1.422 & 1.065 & 1.388 \\ 0.149 & 0.835 & 0.476 & 3.605 & 0.462 & 1.998 \end{pmatrix} \tag{7.2}$$

7.1.3. Discussion

Overall, it was shown that new features decreased *TC* for the given transcriber cohort. However, some discrepancies in efficiency and interaction pattern results need further investigation. It is especially relevant to research why costs for version 3 increased. For this, a closer look at changes of interaction type ratios over multiple versions is taken. Discussion about how past tool evaluation methodologies compare to the log analysis attempted in this thesis is also carried out. Furthermore, an estimate of monetary savings, resulting from the achieved cost reduction, is presented. In the end, threats to validity of the analysis and countermeasures taken to prevent them are elaborated.

Speaker Identification and Passive Interactions

Ratio changes of passive interactions seem to be connected to speaker identification ratios. When the amount of time used to identify speakers increased, passive time also decreased in return. This can be explained due to several reasons.

First, determining if transcriber inactivity was actually devoted to identifying a speaker using another browser window is difficult. Time spent outside the transcription screen was only attributed to speaker identification if elements belonging to speaker selection, speaker search, or profile modification were clicked after coming back.

Secondly, interaction ratio results are influenced by existence of relevant speaker profile pictures in the DDDB. If officials participating in a task's hearing have pictures assigned to them, they can be viewed using the profile picture preview introduced in version 1. Opening and closing the preview is observed by logs and can be attributed to the speaker identification parameter. However, transcribers must search for speaker names on third-party sites if no pictures exist. This presents another potential reason for a gain in passive interaction time.

Furthermore, passive interactions were used as a "catch all" parameter to collect time which could not be attributed to any other interaction type. Results could also be skewed because of this.

In addition, time necessary for speaker identification is highly dependent on state. As seen in Figure 7.3, New York tasks are more expensive than others in version 3. Transcribers might have had problems identifying legislators and persons unknown to them. New York hearing videos also provide the worst video and sound quality, further complicating speaker identification.

Merging, Splitting, and Version 3 Efficiency

Next, the drastic changes in merge and split operations as well as the cost increase for version 3 was investigated. It was discovered that a configuration change in the preprocessing pipeline caused these abnormalities. A modified parameter in a preprocessing script led to generation of shorter utterances in tasks. This produced an increase in merge and decrease in split operations. The vast amount of utterances also led to browsers being slowed down by longer tasks, decelerating the transcription process. These findings can be seen as proof that logs can also be utilized to detect software defects and unexpected behavior.

On a side note, a decline in splitting operations also led to a decrease in text corrections. Basically, sentence boundaries must be modified nearly every time an utterance is split. Therefore, less text correction time can be explained by fewer splitting operations occurring.

As a result of these findings, another efficiency analysis excluding tasks from New York and those containing more than 15 minutes of video was performed. For this analysis of 2,237 tasks (158 hours of video), Table 7.3 shows a 14.80% cost decrease from first to last version. Figure 7.7 visualizes *TC* over all versions for this filtered data. Relation of each version's performance to that of version 0 can be observed in Figure 7.8.

Table 7.3.: Transcription cost results for filtered cohort data.

Version	*TC*	***TC* Change**	
		Vs. Version 0	**Vs. Previous**
0	4.168	-	-
1	3.934	5.59%	5.59%
2	3.705	11.11%	5.84%
3	3.604	13.52%	2.71%
4	3.551	14.80%	1.48%

Using these values, assumptions could be confirmed that side effects were produced by non-ideal preprocessing parameters. Excluding longer tasks also led to an increase in *TC*. This could be explained by a number of reasons. First, transcribers can fast-forward through longer speeches using video features introduced in version 2. Secondly, long testimonies and debates between legislators do not influence factors such as speaker identification. Naturally, speakers only have to be identified once, no matter how often they appear in a video.

Monetary Savings

As presented in Section 7.1.1, version 0 produced a *TC* of 3.935 (3 minutes 56 seconds). Using version 4, experienced transcribers managed to achieve a *TC* of only 3.172 minutes (3 minutes 10 seconds), reducing costs by 19.4%.

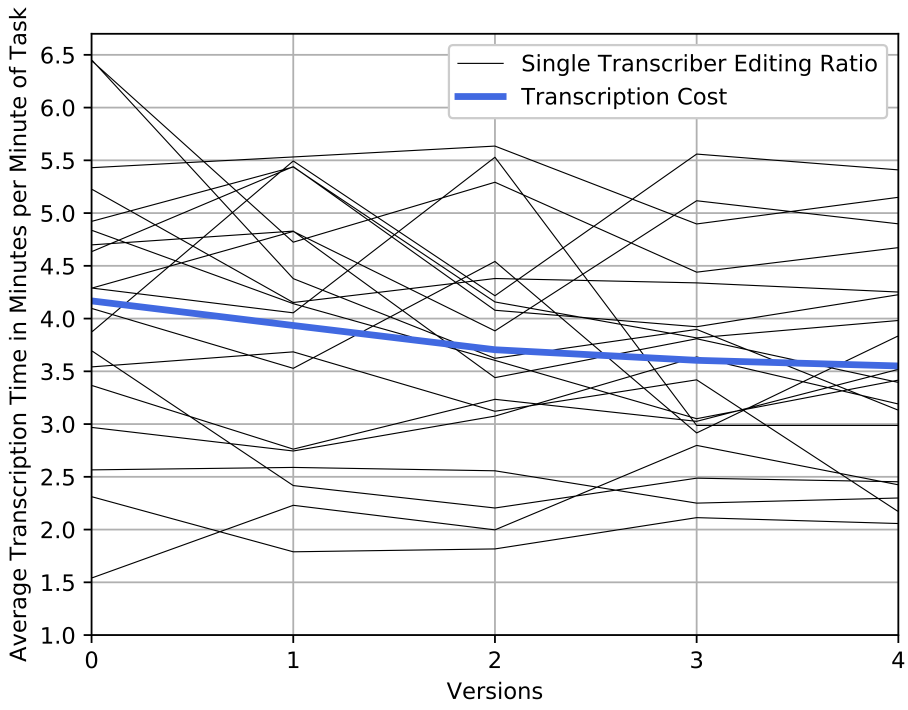


Figure 7.7.: Transcription cost per version for filtered cohort data.

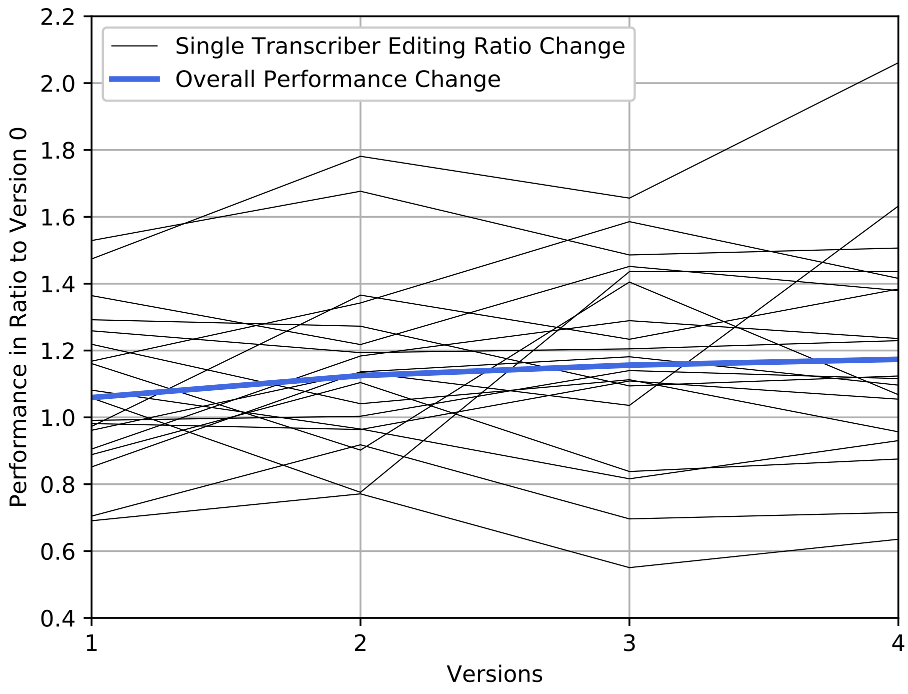


Figure 7.8.: Filtered cohort performance change per version in comparison to version 0.

Assuming an hourly labor rate of $10, such a reduction equates to cost savings of $7.63 per hour of video speech. In March 2018, the Digital Democracy initiative created transcriptions for 545.17 hours of hearing video speech. Applying the presented estimations, a claim can be made that $4159.65 were saved for this one month alone. This would result into a yearly benefit of nearly $50,000 for the initiative, assuming hearing video input would stay similar for all months.

Comparison To Rovin's Evaluation

In contrast to Rovin's evaluation of transcription cost mentioned in Section 3.5.2, no hypothesis regarding transcription quality can be constructed. Log analysis is unfit to detect mistakes regarding speaker assignments or transcription texts for arbitrary tasks. Such an evaluation has to be performed in a controlled environment, where an immaculate test set of tasks is available.

On the other hand, log analysis provides better insight into actual transcription cost and tool efficiency. While helpful for an initial evaluation, lab tests using a small number of transcribers and tasks do not capture real-world working conditions.

Due to new developments being implemented after publication of Rovin's work, results differ vastly. Back in 2016, Rovin's improvements led to an average transcription time of 6.317 minutes (6 minutes 19 seconds) per minute of video, increasing efficiency by 16.89% to when he first started his work. Since TC considers video speech time instead of video length, Rovin's results are normalized to 5.441 for a comparison. This value is calculated by using the average proportion of VS_t to video length as presented in Section 6.1 (86.13%).

Table 7.4 shows a short comparison between the characteristics of Rovin's tool evaluations and the study conducted for this thesis.

Table 7.4.: Comparison between Rovin's tool evaluation and the study conducted in this work. Rovin's results were normalized using average VS_t.

Characteristics	**Rovin**	**Ruprechter**
Transcription Time per Minute of Video Speech	5.441 Minutes	3.935 Minutes
Accomplished Efficiency Improvement	16.89%	19.40%
Number of Transcribers for Evaluation	15	20
Number of Tasks for Evaluation	5	> 3,200
Text Quality Evaluation	Yes	No
Evaluation Method	Lab Study	Log Analysis
Date of Data Collection	Spring 2016	October 2017 – March 2018

Threats to Validity

The presented analysis can only be considered valid if the majority of interaction and efficiency changes are indeed caused by tool improvements. Side effects and non-measurable factors can occur in a real-world testing scenario as featured in this

Listing 7.1: MySQL query for retrieving changes in text areas.

```
SELECT
   l.tagId,
   GROUP_CONCAT(l.value ORDER BY timestamp SEPARATOR '|->|')
      AS changes,
   l.taskId
FROM TT_Log l
WHERE l.element = 'TEXTAREA'
      AND (l.event = 'focusin' OR l.event = 'change')
GROUP BY l.tagId;
```

research. If such factors had major influence on results, validity of the presented analysis would be threatened.

Minimizing effects of unexpected threats was addressed by choosing a large task subset to evaluate tool versions. Although tasks are vastly different, analyzing over 3,200 of them aims at compensating for outliers in the data.

A major threat to validity of the analysis could be the different skill levels of transcribers. Choosing a cohort of 20 experienced transcribers who worked with all versions of the tool mitigates this. By this selection, it can be assumed that all test participants possess a similar skill level.

Furthermore, states strongly influence nature of transcription tasks. Although not directly addressed in the initial analysis in Section 7.1.1, results for separate states were also computed to confirm this. Another possibility of solving this issue was presented in Section 7.1.3, where cohort data was filtered to exclude long tasks and New York hearings.

7.2. Transcription Text Correction

The following few sections describe application of basic NLP techniques to collected log data as elaborated in Section 6.3.

7.2.1. Common Corrections

In order to calculate common replacements, changes in text areas had to be pulled from the DDDB. To achieve this, `focusin` and `change` events triggered by text areas were collected from the `TT_Log` table and then post-processed using Python3. Listing 7.1 shows the query which fetches changes to text areas from the DDDB. It separates before and after values by the String sequence `|->|`.

After processing query results using `NLTK`, `difflib.unified_diff` was run to detect modifications of texts. To properly represent changes, `difflib` output was converted into tuples of removed and added text. Finally, replacements were counted and a readable representation of common corrections was created. This enables researchers to gain a first overview of the data.

Table 7.5 displays results of these replacement computations. It lists the most common corrections and how frequently they occurred.

When looking at the results, it became evident that correction of punctuation was one of the main types of edits performed by transcribers. Due to the huge amount of split and merge operations necessary during transcription, this had to be expected. Automatic algorithms might also wrongly separate sentences when a person pauses while speaking.

In addition to that, transcribers frequently correct people's titles. For example, "Mister" was often replaced by "Mr." and vice versa. According to project administrators, using abbreviations is preferable most of the time. Therefore, personal titles are a candidate for automated correction during preprocessing.

Next, political titles in legislative hearings are often falsely capitalized. Capitalization of words such as Chair, Senator, Secretary, or Representative entirely depends on the context. These words would not necessarily have to be capitalized when used in a normal sentence, but should be when used as titles.

Besides titles, other capitalization errors during the automatic transcription phase exist. It seems like some words are always capitalized or even converted to upper case (e. g. "Learned", "CAVEAT"). Preprocessing scripts have to be corrected to properly address this problem.

Furthermore, transcribers have to frequently remove sound tags created by the automatic transcription service ("[INAUDIBLE]", "[UNKOWN]", etc.). Such corrections seem unautomatable at the current moment.

In some cases, symbols such as dollar, percent, or euro signs are supposed to be represented by their textual representation instead. Again, corrections like that depend on the exact context but could still be automated by use of NLP techniques.

One of the most interesting findings is the common replacement of specific names, especially legislator last names. Some had to be corrected nearly every single time they occurred, for example "McCarty", "McGuire", "Beall", or "Bettencourt". Section 7.2.2 investigates this issue further.

Lastly, the remainder of corrections can be explained due to homophones, wrong tenses, or other corner cases.

7.2.2. Legislator Name Correction

Proofreading the 20 transcription texts in the task test set confirmed that transcribers frequently miss misspelled names. While on average only one name was misspelled for completed Texas tasks, other states show higher values. For California, two tasks contained three names which were still misspelled. A single Florida task even showed seven misspelled names. Overall, only two of the twenty test tasks could be considered immaculate when looking at spelling of names. This proved the assumption that an experiment for text correction is only feasible using proofread texts.

Using these proofread texts, results for dictionary replacement as well as NLP

Table 7.5.: The most common transcriber corrections and their frequency of occurrence.

Added	Removed	#
	,	17280
,		5890
. And	, and	4179
.	?	4090
[INAUDIBLE]		3272
. And	and	3182
.	,	2701
	.	2546
,	.	2006
. But	, but	1712
.		1654
,	?	1482
	$	1176
Mr	Mr.	1129
members	Members	1128
state	State	1087
[UNKNOWN]		1035
And	and	879
Thank you.		758
Yes.		733
bill	Bill	724
Committee	committee	656
?	.	595
	the	571
And		534
Okay.		520
Learned	learned	481
Aye,		457
because	Because	443
	And	437
	a	436
the		396
Reserve	reserve	393
substitute	Substitute	392
	Thank you.	381
	and	335
	that	324
Right.		320
. Because	because	318
Aye.		318
. The	, the	300
%	percent	300
$		299
. So	, so	296
The	the	294
Doctor	doctor	292
[INAUDIBLE]	...	282
	-	281
Yeah .		280
[UNKNOWN]	...	279
. We	, we	279
secretary	Secretary	264
	”	263
	of	261
That	that	258
	dollars	254
Bill	bill	253
. To	to	253
. That	that	253
, and	. And	249
	to	241
, we	. We	240
and	in	238
. That	, that	235
. Which	, which	234
	Yes.	234
Mm-hm.		228
	?	227
, the	. The	227
a		224
Mister	Mr.	223
.	...	218
house	House	218
is	’s	217
	...	215
	“	214
So		211
amendment	Amendment	206

approaches using Levenshtein and Jaro-Winkler distances were computed.

Dictionary Replacement

Because of common correction results presented in Section 7.2.1 above, additions were made to dictionary prerequisites defined in Section 6.3.2. Corrections of sound tags (e. g. "[INAUDIBLE]", "[UNKNOWN]") were omitted, since they are always subject to replacement. Attempting to automatically replace such tags would lead to faulty corrections. Furthermore, the dictionary excludes replacements containing terms such as "Aye", "Naye", "No", or "Yes". This vocabulary is frequently used during votes or roll calls. Such utterances are subject to excessive changes caused by split and merge operations. Including these would distort dictionary content.

When looking at the results for the common replacement dictionary, it is observable that only minor corrections were carried out. On average, less than one replacement could be performed per task. The dictionary had no impact at all for New York and Texas. For the chosen Florida tasks, 3 replacements could be made on average. Although correction numbers are low, all of the performed replacements were correct. Table 7.6 shows dictionary replacement results, averaged over all states and tasks.

Table 7.6.: Dictionary replacement results.

State	$\Delta_{u,c}$	ϵ	m	c
CA	0.6	0	5.4	0.6
FL	3	0	2.2	3
NY	0	0	4.2	0
TX	0	0	1.6	0
Overall	0.9	0	3.35	0.9

Automatic Name Correction

As explained in Section 6.3.2, evaluation was carried out for replacements using capitalization as well as POS- and NER-taggers provided by Stanford and spaCy. Levenshtein and Jaro-Winkler algorithms were used to determine word distance. Therefore, multiple experiment runs were first carried out to find appropriate values for θ_{Lr} and θ_{JW}. According to Section 6.3.2, threshold ranges were chosen as $0.23 \leq \theta_{Lr} \leq 0.27$ and $0.86 \leq \theta_{JW} \leq 0.9$. Full results for these experiments can be observed in Tables A.1 and A.2 in the Appendix.

After evaluating results for different θ_{Lr}, any value between 0.25 and 0.27 seemed acceptable. These thresholds all perform equally in consideration of the evaluated tasks. However, results could prove to be misleading given the relatively small test set. To prevent possible false corrections for different tasks, $\theta_{Lr} = 0.25$ should be considered favorable in contrast to higher values. Table 7.7 lists overall results and averages for the evaluated θ_{Lr} threshold range.

Investigating results of algorithms using Levenshtein distance, the approach utilizing spaCy's NER-tagger was the only one producing no false corrections. On the other hand, it missed 3 misspelled names while correcting about 1.25 names per task on average. Simple capitalization replacement also performed well, producing an error ratio of $E = 0.034$ on average while surprisingly also correctly replacing 1.4 names. This fact combined with $\Delta_{u,c} = 1.45$ for capitalization means it proved to be the most effective correction algorithm using Levenshtein. Disappointingly, the approach utilizing Stanford's NER-tagger produced the worst results for all metrics.

Table 7.7.: Overall results and averages for Levenshtein threshold computations.

Approaches	**Metrics**	**Threshold Range**				
		0.27	**0.26**	**0.25**	**0.24**	**0.23**
Capitalization	$\Delta_{u,c}$	1.45	1.45	**1.45**	1.35	1.35
	m	2.85	2.85	**2.85**	2.95	2.95
	ϵ	0.05	0.05	**0.05**	0.05	0.05
	c	1.4	1.4	**1.4**	1.3	1.3
	E	0.034	0.034	**0.034**	0.037	0.037
POS Stanford	$\Delta_{u,c}$	1.4	1.4	**1.4**	1.3	1.3
	m	2.9	2.9	**2.9**	3	3
	ϵ	0.05	0.05	**0.05**	0.05	0.05
	c	1.35	1.35	**1.35**	1.25	1.25
	E	0.036	0.036	**0.036**	0.038	0.038
NER Stanford	$\Delta_{u,c}$	1.1	1.1	**1.1**	1	1
	m	3.2	3.2	**3.2**	3.3	3.3
	ϵ	0.05	0.05	**0.05**	0.05	0.05
	c	1.05	1.05	**1.05**	0.95	0.95
	E	0.045	0.045	**0.045**	0.05	0.05
POS spaCy	$\Delta_{u,c}$	1.25	1.25	**1.25**	1.15	1.15
	m	3.05	3.05	**3.05**	3.15	3.15
	ϵ	0.05	0.05	**0.05**	0.05	0.05
	c	1.2	1.2	**1.2**	1.1	1.1
	E	0.04	0.04	**0.04**	0.043	0.043
NER spaCy	$\Delta_{u,c}$	1.25	1.25	**1.25**	1.2	1.2
	m	3	3	**3**	3.05	3.05
	ϵ	0	0	**0**	0	0
	c	1.25	1.25	**1.25**	1.2	1.2
	E	0	0	**0**	0	0
Average	$\Delta_{u,c}$	1.29	1.29	**1.29**	1.2	1.2
	m	3	3	**3**	3.09	3.09
	ϵ	0.04	0.04	**0.04**	0.04	0.04
	c	1.25	1.25	**1.25**	1.16	1.16
	E	0.031	0.031	**0.031**	0.034	0.034

For Jaro-Winkler, $\theta_{JW} = 0.88$ was chosen as the best-fitting configuration. Through all replacement approaches, this threshold performed best considering the observed parameters. Table 7.8 shows overall results for the investigated Jaro-Winkler threshold range. When using $\theta_{JW} = 0.88$, an error rate of 0.026 was produced on average, while 1.55 misspelled names were missed and 2.7 names were correctly replaced.

Simple capitalization can be considered infeasible using $\theta_{JW} = 0.88$, because $E = 0.077$ represents a considerably higher value than other approaches having similar $\Delta_{u,c}$. For the approach utilizing spaCy's NER-tagger, E reached 0. However, not as many corrections were made than with other approaches ($\Delta_{u,c} = 2.3$). Out of all algorithms making use of Jaro-Winkler distance, the one using Stanford's POS-tagger performed best. It generated nearly 3 correct replacements on average and missed only 1.3 misspelled names per task, while producing an error rate of 0.017.

7.2.3. Discussion

Multiple conclusions can be drawn from the computed list of common replacements.

After talking to transcribers and project leaders about common corrections as presented in Section 7.2.1, it became obvious that correction guidelines were not always well-defined. This led to uncertainty among transcribers which resulted in faulty text corrections. Specifying clear policies might help to remove these ambiguities. In addition, context-dependent capitalization of legislative vocabulary (Bill, Senator, Member, etc.), abbreviation of people's titles, and textual representation of specific symbols might be eligible for automated correction.

When discussing automatic legislator name correction, it can be stated that simple replacement approaches as presented in Section 7.2.2 only produce a limited number of corrections. This is explainable due to the conservative choice of threshold values, which focuses on reducing errors. Although an adjustment to thresholds could increase $\Delta_{u,c}$ and decrease m, it would also increase ϵ and E and therefore reduce text quality.

The small task set presents a threat to the validity of the attempted evaluation. However, there currently exists no effective way of comparing automatic name replacements in completed task texts without immaculate transcripts.

To conduct an evaluation which has access to more data, A/B/n testing could be performed. Tasks could be separated into groups, with each replacement algorithm applied to a different task set. Then, text correction time and word replacements could be monitored to assess if correction frequency changes. In addition, a smaller task set could be selected to manually evaluate text quality using $\Delta_{t,p}$.

Table 7.8.: Overall results and averages for Jaro-Winkler threshold computations.

Approaches	**Metrics**	**Threshold Range**				
		0.86	**0.87**	**0.88**	**0.89**	**0.9**
Capitalization	$\Delta_{u,c}$	3.4	3.35	**3.25**	3.2	2.9
	m	1.15	1.2	**1.25**	1.3	1.6
	ϵ	0.3	0.3	**0.25**	0.25	0.25
	c	3.1	3.05	**3**	2.95	2.65
	E	0.088	0.09	**0.077**	0.078	0.086
POS Stanford	$\Delta_{u,c}$	3.1	3.1	**3**	2.95	2.65
	m	1.25	1.25	**1.3**	1.35	1.65
	ϵ	0.1	0.1	**0.05**	0.05	0.05
	c	3	3	**2.95**	2.9	2.6
	E	0.032	0.032	**0.017**	0.017	0.019
NER Stanford	$\Delta_{u,c}$	2.65	2.65	**2.55**	2.5	2.25
	m	1.7	1.7	**1.75**	1.8	2.05
	ϵ	0.1	0.1	**0.05**	0.05	0.05
	c	2.55	2.55	**2.5**	2.45	2.2
	E	0.038	0.038	**0.02**	0.02	0.022
POS spaCy	$\Delta_{u,c}$	2.9	2.9	**2.8**	2.75	2.45
	m	1.45	1.45	**1.5**	1.55	1.85
	ϵ	0.1	0.1	**0.05**	0.05	0.05
	c	2.8	2.8	**2.75**	2.7	2.4
	E	0.034	0.034	**0.018**	0.018	0.02
NER spaCy	$\Delta_{u,c}$	2.35	2.35	**2.3**	2.3	2.05
	m	1.9	1.9	**1.95**	1.95	2.2
	ϵ	0	0	**0**	0	0
	c	2.35	2.35	**2.3**	2.3	2.05
	E	0	0	**0**	0	0
Average	$\Delta_{u,c}$	2.88	2.87	**2.78**	2.74	2.46
	m	1.49	1.5	**1.55**	1.59	1.87
	ϵ	0.12	0.12	**0.08**	0.08	0.08
	c	2.76	2.75	**2.7**	2.66	2.38
	E	0.038	0.039	**0.026**	0.027	0.029

7.3. Summary

This chapter presented results for metrics defined in Chapter 6, including efficiency analysis, transcriber interaction patterns, common text corrections, and automatic name replacements.

An efficiency analysis conducted for a cohort of 20 transcribers which completed over 3,200 tasks led to conclusion that the implemented tool improvements reduced *TC* by 19.4% over four versions. Using version 4, the average time necessary to transcribe one minute of video speech (*TC*) was 3.172, in contrast to 3.935 for baseline version 0. It was further shown that results are strongly influenced by factors such as state, with New York and California tasks producing higher *TC* in version 3.

Interaction analysis proved that assumptions of administrators about composition of transcription time were mostly correct. Speaker identification and passive time took up most of a transcriber's time, while text correction had less of an influence. In contrast, startup time was surprisingly high. The number of split and merge operations fluctuated over different versions, with respective values strongly changing after version 3. This could be backtracked to unexpected results of preprocessing pipeline adjustments, demonstrating that interaction analysis is also useful for fault detection.

Next, findings for common replacements performed by transcribers were presented. The most common sources of text correction were identified as: punctuation and sentence boundaries, abbreviated titles, capitalization of legislative vocabulary, transcription service sound tags, symbols, and legislator names.

Lastly, an evaluation of automatic legislator name replacements was carried out. Through all approaches, only a small number of text corrections was performed. Although not introducing errors in text, dictionary replacement merely produced 0.9 corrections per task. More sophisticated algorithms based on POS-tagging, NER-tagging, and capitalization behaved quite similar when compared to each other. While NLP approaches utilizing Jaro-Winkler distance accomplished 2.95 correct name replacements per task on average, those using Levenshtein distance only achieved 1.4 replacements. Threshold values could be adjusted to produce more corrections, but even small changes could negatively impact text quality.

8. Conclusion and Future Work

This Chapter summarizes the contributions performed throughout this thesis and provides an outlook on possible future improvements and research.

8.1. Conclusion

In the absence of fully automated, highly accurate transcription technology, human-assisted transcription is at present a viable and cost effective option for capturing speech from legislative proceedings. This thesis focused on building a deeper understanding of human-assisted transcription systems, how to make them more efficient, and finally how the human contribution is distributed across various interaction types expected from transcribers. Gaining understanding in the transcription process was achieved by examining results of the implemented logging system. To further prove usefulness of this work, basic text analysis using natural language processing was carried out.

In this thesis, the Digital Democracy initiative and the Transcription Tool necessary for the bulk of input data preparation for this project were introduced. A technology rework and four sets of improvements were implemented for this tool, which aimed at increasing overall efficiency. Improvements were sequentially released as four separate tool versions. While baseline version 0 represented the developed technology rework, versions 1 to 4 introduced new features. Version 1 incorporated speaker profile preview into the tool. In version 2, enhanced video player functionality was implemented. Version 3 enabled better utterance navigation and interactivity with the transcription screen. Finally, version 4 connected the Transcription Tool to an automated speaker identification service implemented in another student project.

Furthermore, a logging system was developed to allow sophisticated analysis of the transcription process. This system collected data for over 12,000 transcription tasks and about 2,300 hours of video, resulting in roughly 8,000 hours of transcription work.

Using the collected logs, a study of how tool improvements affected efficiency and cost of the entire operation was performed. A cohort of 20 transcribers was selected for this analysis. Transcription cost (*TC*) was defined as the main metric measuring tool efficiency. *TC* stands for the average time it takes a human transcriber to transcribe one minute of video speech using the Transcription Tool. It was found that on average a 19.4% decrease in *TC*, and therefore increase in efficiency, could be realized over four versions. During this analysis, it was also shown that side effects and factors such as state, preprocessing parameters, and task duration influence the transcription process and consequently TC.

To analyze how human transcribers interact with the Transcription Tool, transcription time was split up into six interaction types in this thesis. The following interaction types were investigated: speaker identification, text correction, tool startup, splitting and merging utterances, as well as the transcriber being idle (passive). By utilizing log data, these interactions were quantified to assess their impact on the overall transcription process. It was confirmed that speaker identification and passive time accounted for over 65% of combined transcription time in all versions. Text correction only took up around 12-15% of time on average, meaning that automatic transcription services produce good-quality texts. Splitting and merging operations are heavily dependent on preprocessing parameters such as utterance length. Startup time was higher than expected, reaching close to 10% for nearly all versions.

Lastly, a proof of concept demonstrating that log data can be used for more advanced analysis such as NLP was performed. Common transcriber corrections were collected to gain insight into text modifications. From this, sources of corrections were classified and improvement possibilities were laid out. One of these sources, misspelled legislator names, was investigated further. Text analysis utilizing POS- and NER-tagging as well as Levenshtein and Jaro-Winkler distance algorithms was carried out. Although only an average of about 3 automated corrections per task were performed for the selected test set, feasibility of this research was proven.

8.2. Future Work

Nearly all topics explored in this thesis could serve as a basis for future work and research.

Many possibilities exist for new Transcription Tool features. One of the long-planned improvements is a widget which enables transcribers to automatically add profile pictures for the current speaker to the database. Using the present video frame, transcribers could create snapshots of a person's face. While such a tool could slow down the transcription process initially, it should increase both transcription quality and speed in the long run.

Automatic speaker identification is currently only available for legislators. It might be relevant to enable speaker suggestions for frequently appearing non-officials such as lobbyists.

In addition to speaker suggestion implemented for this work, suggestions for organization affiliation of speakers could be relevant. Although it was not a major focus of this thesis, identifying which organization a speaker represents is time-consuming. A more sophisticated organization selection dialog could speed up organization search and prevent wrong affiliation assignments.

Since splitting utterances is a tedious task, it is important to find preprocessing parameters which produce utterances of correct length. In this regard, current configuration is still flawed. Transcribers also have to manually adjust utterance start and end time for every split. If there was a way to estimate the time for which a split operation was triggered, correction overhead could be decreased. Furthermore, warning messages could be introduced which indicate if utterances need to be split or merged. This would reduce need for transcribers to evaluate utterance length manually.

For text and log analysis, a wide range of possibilities for future research exists. Log data produced by the implemented system could be utilized for further analysis, regarding both efficiency metrics and interaction types. In the efficiency evaluation attempted for this thesis, only experienced transcribers were considered. In the future, it might be interesting to assess performance of new transcribers. Comparisons could be attempted regarding learning curves in different tool versions. As an example, investigating how a transcriber's speaker identification time changes over time could indicate time necessary to familiarize with speakers in specific states and committees. Automatic outlier detection could also be utilized to detect transcribers abusing the Digital Democracy initiative by not completing transcription tasks in acceptable time.

Although basic NLP concepts were explored, this thesis refrained from doing research into neural networks and deep learning approaches. Such advanced technologies could produce interesting results. Research might prove to be fruitful for multiple purposes such as spell checking, providing support to human transcribers during transcription, or prediction of future tool efficiency and defects.

Appendix A.

Computation Results for Automatic Name Correction

Table A.1.: Complete results for Levenshtein threshold computations.

θ_{JW}	Task Set	Capitalization				POS Stanford				NER Stanford				POS spaCy				NER spaCy			
		$\Delta_{u,c}$	m	ϵ	c	$\Delta_{u,c}$	m	ϵ	c	$\Delta_{u,c}$	m	ϵ	c	$\Delta_{u,c}$	m	ϵ	c	$\Delta_{u,c}$	m	ϵ	c
0.27	CA	2.8	3.2	0	2.8	2.8	3.2	0	2.8	2	4	0	2	2.4	3.6	0	2.4	3.2	2.8	0	3.2
	FL	1.2	4.2	0.2	1	1	4.4	0.2	0.8	0.6	4.8	0.2	0.4	0.8	4.6	0.2	0.6	0.4	4.8	0	0.4
	NY	1.4	2.8	0	1.4	1.4	2.8	0	1.4	1.4	2.8	0	1.4	1.4	2.8	0	1.4	1	3.2	0	1
	TX	0.4	1.2	0	0.4	0.4	1.2	0	0.4	0.4	1.2	0	0.4	0.4	1.2	0	0.4	0.4	1.2	0	0.4
	Overall	1.45	2.85	0.05	1.4	1.4	2.9	0.05	1.35	1.1	3.2	0.05	1.05	1.25	3.05	0.05	1.2	1.25	3	0	1.25
0.26	CA	2.8	3.2	0	2.8	2.8	3.2	0	2.8	2	4	0	2	2.4	3.6	0	2.4	3.2	2.8	0	3.2
	FL	1.2	4.2	0.2	1	1	4.4	0.2	0.8	0.6	4.8	0.2	0.4	0.8	4.6	0.2	0.6	0.4	4.8	0	0.4
	NY	1.4	2.8	0	1.4	1.4	2.8	0	1.4	1.4	2.8	0	1.4	1.4	2.8	0	1.4	1	3.2	0	1
	TX	0.4	1.2	0	0.4	0.4	1.2	0	0.4	0.4	1.2	0	0.4	0.4	1.2	0	0.4	0.4	1.2	0	0.4
	Overall	1.45	2.85	0.05	1.4	1.4	2.9	0.05	1.35	1.1	3.2	0.05	1.05	1.25	3.05	0.05	1.2	1.25	3	0	1.25
0.25	CA	2.8	3.2	0	2.8	2.8	3.2	0	2.8	2	4	0	2	2.4	3.6	0	2.4	3.2	2.8	0	3.2
	FL	1.2	4.2	0.2	1	1	4.4	0.2	0.8	0.6	4.8	0.2	0.4	0.8	4.6	0.2	0.6	0.4	4.8	0	0.4
	NY	1.4	2.8	0	1.4	1.4	2.8	0	1.4	1.4	2.8	0	1.4	1.4	2.8	0	1.4	1	3.2	0	1
	TX	0.4	1.2	0	0.4	0.4	1.2	0	0.4	0.4	1.2	0	0.4	0.4	1.2	0	0.4	0.4	1.2	0	0.4
	Overall	1.45	2.85	0.05	1.4	1.4	2.9	0.05	1.35	1.1	3.2	0.05	1.05	1.25	3.05	0.05	1.2	1.25	3	0	1.25
0.24	CA	2.8	3.2	0	2.8	2.8	3.2	0	2.8	2	4	0	2	2.4	3.6	0	2.4	3.2	2.8	0	3.2
	FL	1.2	4.2	0.2	1	1	4.4	0.2	0.8	0.6	4.8	0.2	0.4	0.8	4.6	0.2	0.6	0.4	4.8	0	0.4
	NY	1	3.2	0	1	1	3.2	0	1	1	3.2	0	1	1	3.2	0	1	0.8	3.4	0	0.8
	TX	0.4	1.2	0	0.4	0.4	1.2	0	0.4	0.4	1.2	0	0.4	0.4	1.2	0	0.4	0.4	1.2	0	0.4
	Overall	1.35	2.95	0.05	1.3	1.3	3	0.05	1.25	1	3.3	0.05	0.95	1.15	3.15	0.05	1.1	1.2	3.05	0	1.2
0.23	CA	2.8	3.2	0	2.8	2.8	3.2	0	2.8	2	4	0	2	2.4	3.6	0	2.4	3.2	2.8	0	3.2
	FL	1.2	4.2	0.2	1	1	4.4	0.2	0.8	0.6	4.8	0.2	0.4	0.8	4.6	0.2	0.6	0.4	4.8	0	0.4
	NY	1	3.2	0	1	1	3.2	0	1	1	3.2	0	1	1	3.2	0	1	0.8	3.4	0	0.8
	TX	0.4	1.2	0	0.4	0.4	1.2	0	0.4	0.4	1.2	0	0.4	0.4	1.2	0	0.4	0.4	1.2	0	0.4
	Overall	1.35	2.95	0.05	1.3	1.3	3	0.05	1.25	1	3.3	0.05	0.95	1.15	3.15	0.05	1.1	1.2	3.05	0	1.2

Table A.2.: Complete results for Jaro-Winkler threshold computations.

θ_{JW}	Task Set	Capitalization				POS Stanford				NER Stanford				POS spaCy				NER spaCy			
		$\Delta_{u,c}$	m	ϵ	c	$\Delta_{u,c}$	m	ϵ	c	$\Delta_{u,c}$	m	ϵ	c	$\Delta_{u,c}$	m	ϵ	c	$\Delta_{u,c}$	m	ϵ	c
0.86	CA	5.4	1.2	0.6	4.8	4.8	1.2	0	4.8	3.8	2.2	0	3.8	4.2	1.8	0	4.2	4.8	1.2	0	4.8
	FL	4.2	1.2	0.2	4	3.8	1.6	0.2	3.6	3.2	2.2	0.2	3	3.6	1.8	0.2	3.4	1.8	3.4	0	1.8
	NY	2.8	1.8	0.4	2.4	2.6	1.8	0.2	2.4	2.6	1.8	0.2	2.4	2.6	1.8	0.2	2.4	1.8	2.4	0	1.8
	TX	1.2	0.4	0	1.2	1.2	0.4	0	1.2	1	0.6	0	1	1.2	0.4	0	1.2	1	0.6	0	1
	Overall	3.4	1.15	0.3	3.1	3.1	1.25	0.1	3	2.65	1.7	0.1	2.55	2.9	1.45	0.1	2.8	2.35	1.9	0	2.35
0.87	CA	5.4	1.2	0.6	4.8	4.8	1.2	0	4.8	3.8	2.2	0	3.8	4.2	1.8	0	4.2	4.8	1.2	0	4.8
	FL	4	1.4	0.2	3.8	3.8	1.6	0.2	3.6	3.2	2.2	0.2	3	3.6	1.8	0.2	3.4	1.8	3.4	0	1.8
	NY	2.8	1.8	0.4	2.4	2.6	1.8	0.2	2.4	2.6	1.8	0.2	2.4	2.6	1.8	0.2	2.4	1.8	2.4	0	1.8
	TX	1.2	0.4	0	1.2	1.2	0.4	0	1.2	1	0.6	0	1	1.2	0.4	0	1.2	1	0.6	0	1
	Overall	3.35	1.2	0.3	3.05	3.1	1.25	0.1	3	2.65	1.7	0.1	2.55	2.9	1.45	0.1	2.8	2.35	1.9	0	2.35
0.88	CA	5.4	1.2	0.6	4.8	4.8	1.2	0	4.8	3.8	2.2	0	3.8	4.2	1.8	0	4.2	4.8	1.2	0	4.8
	FL	3.8	1.6	0.2	3.6	3.6	1.8	0.2	3.4	3	2.4	0.2	2.8	3.4	2	0.2	3.2	1.6	3.6	0	1.6
	NY	2.6	1.8	0.2	2.4	2.4	1.8	0	2.4	2.4	1.8	0	2.4	2.4	1.8	0	2.4	1.8	2.4	0	1.8
	TX	1.2	0.4	0	1.2	1.2	0.4	0	1.2	1	0.6	0	1	1.2	0.4	0	1.2	1	0.6	0	1
	Overall	3.25	1.25	0.25	3	3	1.3	0.05	2.95	2.55	1.75	0.05	2.5	2.8	1.5	0.05	2.75	2.3	1.95	0	2.3
0.89	CA	5.4	1.2	0.6	4.8	4.8	1.2	0	4.8	3.8	2.2	0	3.8	4.2	1.8	0	4.2	4.8	1.2	0	4.8
	FL	3.8	1.6	0.2	3.6	3.6	1.8	0.2	3.4	3	2.4	0.2	2.8	3.4	2	0.2	3.2	1.6	3.6	0	1.6
	NY	2.4	2	0.2	2.2	2.2	2	0	2.2	2.2	2	0	2.2	2.2	2	0	2.2	1.8	2.4	0	1.8
	TX	1.2	0.4	0	1.2	1.2	0.4	0	1.2	1	0.6	0	1	1.2	0.4	0	1.2	1	0.6	0	1
	Overall	3.2	1.3	0.25	2.95	2.95	1.35	0.05	2.9	2.5	1.8	0.05	2.45	2.75	1.55	0.05	2.7	2.3	1.95	0	2.3
0.9	CA	5	1.6	0.6	4.4	4.4	1.6	0	4.4	3.4	2.6	0	3.4	3.8	2.2	0	3.8	4.4	1.6	0	4.4
	FL	3.8	1.6	0.2	3.6	3.6	1.8	0.2	3.4	3	2.4	0.2	2.8	3.4	2	0.2	3.2	1.6	3.6	0	1.6
	NY	2	2.4	0.2	1.8	1.8	2.4	0	1.8	1.8	2.4	0	1.8	1.8	2.4	0	1.8	1.4	2.8	0	1.4
	TX	0.8	0.8	0	0.8	0.8	0.8	0	0.8	0.8	0.8	0	0.8	0.8	0.8	0	0.8	0.8	0.8	0	0.8
	Overall	2.9	1.6	0.25	2.65	2.65	1.65	0.05	2.6	2.25	2.05	0.05	2.2	2.45	1.85	0.05	2.4	2.05	2.2	0	2.05

Bibliography

Agosti, M., Crivellari, F., & Di Nunzio, G. M. (2012). Web log analysis: a review of a decade of studies about information acquisition, inspection and interpretation of user interaction. *Data Mining and Knowledge Discovery*, *24*(3), 663–696.

Alfred, R., Leong, L. C., On, C. K., & Anthony, P. (2014). Malay named entity recognition based on rule-based approach. *International Journal of Machine Learning and Computing*, *4*(3), 300–306.

Amazon Web Services Inc. (2018). Amazon Simple Storage Service (S3). Retrieved April 21, 2018, from aws.amazon.com/s3

Amazon Web Services, Inc. (2018). Amazon web services - cloud computing services. Retrieved January 15, 2018, from aws.amazon.com

Basu, J., Bepari, M. S., Nandi, S., Khan, S., & Roy, R. (2013, November). SATT: semi-automatic transcription tool. In *2013 international conference oriental COCOSDA held jointly with 2013 conference on asian spoken language research and evaluation (O-COCOSDA/CASLRE)* (pp. 1–6). doi:10.1109/ICSDA.2013.6709903

Berger, A. L., Pietra, V. J. D., & Pietra, S. A. D. (1996). A maximum entropy approach to natural language processing. *Computational linguistics*, *22*(1), 39–71.

Blakeslee, S., Dekhtyar, A., Khosmood, F., Kurfess, F., Kuboi, T., Poschman, H., . . . Durst, S. (2015). Digital Democracy project: making government more transparent one video at a time. *Global Digital Humanities*.

Boehner, K. & DiSalvo, C. (2016). Data, design and civics: an exploratory study of civic tech. In *Proceedings of the 2016 chi conference on human factors in computing systems* (pp. 2970–2981). CHI '16. San Jose, California, USA: ACM. doi:10.1145/2858036.2858326

Brants, T. (2000). TnT: a statistical part-of-speech tagger. In *Proceedings of the sixth conference on applied natural language processing* (pp. 224–231). Association for Computational Linguistics.

Brightcove, Inc. (2018). Video.js: The Player Framework. Retrieved July 10, 2018, from videojs.com

Brill, E. (1995). Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging. *Computational linguistics*, *21*(4), 543–565.

Budhwar, A., Kuboi, T., Dekhtyar, A., & Khosmood, F. (2018). Predicting the vote using legislative speech. In *Proceedings of the 19th annual international conference on digital government research: governance in the data age* (p. 35). ACM.

Burke, M., Amento, B., & Isenhour, P. (2006). Error correction of voicemail transcripts in scanmail. In *Proceedings of the sigchi conference on human factors in* *computing systems* (pp. 339–348). CHI '06. Montreal, Quebec, Canada: ACM. doi:10.1145/1124772.1124823

Busany, N. & Maoz, S. (2016). Behavioral log analysis with statistical guarantees. In *Software engineering (icse), 2016 ieee/acm 38th international conference on* (pp. 877–887). IEEE.

California Polytechnic State University. (2014). Projects - IATPP: Digital Democracy. Retrieved April 25, 2018, from iatpp.calpoly.edu/projects/digitaldemocracy.asp

California Secretary of State. (2017, July). Cal-access. Retrieved April 25, 2018, from cal-access.sos.ca.gov

Castro-Bleda, M. J., Vilar, J. M., Llorens, D., Marzal, A., Prat, F., & Zamora-Martinez, F. (2017). A system for assisted transcription and annotation of ancient documents. In *Proceedings of the 15th international workshop on content-based multimedia indexing* (37:1–37:5). CBMI '17. Florence, Italy: ACM. doi:10.1145/3095713.3095752

Christen, P. (2006). A comparison of personal name matching: techniques and practical issues. In *Data mining workshops, 2006. icdm workshops 2006. sixth ieee international conference on* (pp. 290–294). IEEE.

cielo24. (2018). Cielo24. Retrieved January 10, 2018, from cielo24.com

Clift, S. (2003). E-democracy, e-governance and public net-work. Retrieved July 15, 2018, from publicus.net/articles/edempublicnetwork.html

CNN.com. (2017, February). State politics are the next battleground – and tracking them just got easier. Retrieved September 25, 2018, from money.cnn.com/2017/02/07/technology/digital-democracy-new-york-california-transparency

Cohen, W., Ravikumar, P., & Fienberg, S. (2003). A comparison of string metrics for matching names and records. In *Kdd workshop on data cleaning and object consolidation* (Vol. 3, pp. 73–78).

C-SPAN. (2018). Congressional chronicle - members of congress, hearings and more. Retrieved January 15, 2018, from c-span.org/congress

Cutting, D., Kupiec, J., Pedersen, J., & Sibun, P. (1992). A practical part-of-speech tagger. In *Proceedings of the third conference on applied natural language processing* (pp. 133–140). Association for Computational Linguistics.

Daelemans, W., Zavrel, J., Berck, P., & Gillis, S. (1996). MBT: a memory-based part of speech tagger-generator. *arXiv preprint cmp-lg/9607012*.

Davis, B. & Baxandall, P. (2014, April). Following the money 2014. Retrieved April 25, 2018, from calpirgedfund.org/reports/caf/following-money-2014

Dawes, S. S. & Helbig, N. (2010). Information strategies for open government: challenges and prospects for deriving public value from government transparency. In *International conference on electronic government* (pp. 50–60). Springer.

de Santana, V. F. & Baranauskas, M. C. C. (2015). WELFIT: a remote evaluation tool for identifying web usage patterns through client-side logging. *International Journal of Human-Computer Studies*, *76*, 40–49.

Defazio, A. (2016). A complete guide to the bayes factor test.

Deshpande, R., Tuna, T., Subhlok, J., & Barker, L. (2014, October). A crowdsourcing caption editor for educational videos. In *2014 IEEE frontiers in education conference (FIE) proceedings* (pp. 1–8). doi:10.1109/FIE.2014.7044040

Digital Democracy initiative. (2017a). Digital Democracy homepage. Retrieved April 20, 2018, from digitaldemocracy.org

Digital Democracy initiative. (2017b, November 18). Digitaldemocracy csc iab presentation 2017. Retrieved March 5, 2018, from drive.google.com/open?id=1z-FVDMuSxdCyjqQQSuDb-WutRXrY9AJG

Drupal Association. (2018). Drupal - Open Source CMS. Retrieved June 6, 2018, from drupal.org

Dumais, S., Jeffries, R., Russell, D. M., Tang, D., & Teevan, J. (2014). Understanding user behavior through log data and analysis. In *Ways of knowing in HCI* (pp. 349–372). Springer.

Eftimov, T., Koroušić Seljak, B., & Korošec, P. (2017, June). A rule-based named-entity recognition method for knowledge extraction of evidence-based dietary recommendations. *PLOS ONE*, *12*(6), 1–32. doi:10.1371/journal.pone.0179488

Explosion AI. (2018). spaCy - industrial-strength natural language processing in python. Retrieved July 1, 2018, from aws.amazon.com

Fenstermacher, K. D. & Ginsburg, M. (2002). Mining client-side activity for personalization. In *Advanced issues of e-commerce and web-based information systems, 2002.(wecwis 2002). proceedings. fourth ieee international workshop on* (pp. 205–212). IEEE.

Fernandez, A., Insfran, E., & Abrahão, S. (2011). Usability evaluation methods for the web: a systematic mapping study. *Information and Software Technology*, *53*(8), 789–817.

Fox, J. (2007). Government transparency and policymaking. *Public choice*, *131*(1-2), 23–44.

Gerken, J., Bak, P., Jetter, C., Klinkhammer, D., & Reiterer, H. (2008). How to use interaction logs effectively for usability evaluation.

Ghezzi, C., Pezzè, M., Sama, M., & Tamburrelli, G. (2014). Mining behavior models from user-intensive web applications. In *Proceedings of the 36th international conference on software engineering* (pp. 277–287). ACM.

Gierke, O., Darimont, T., Strobl, C., Paluch, M., & Bryant, J. (2018, July 26). Spring Data JPA - reference documentation. Retrieved August 1, 2018, from docs.spring.io/spring-data/jpa/docs/current/reference/html

GitHub, Inc. (2018). GitHub. Retrieved May 26, 2018, from github.com

Google LLC. (2018). SyntaxNet: Neural Models of Syntax. Retrieved July 20, 2018, from github.com/tensorflow/models/tree/master/research/syntaxnet

Grimes, C., Tang, D., & Russell, D. M. (2007). Query logs alone are not enough. In *Workshop on query log analysis at www*. Citeseer.

Hirschberg, D. S. (1975, June). A linear space algorithm for computing maximal common subsequences. *Communications of the ACM*, *18*(6), 341–343. doi:10.1145/360825.360861

Hong, J. I., Heer, J., Waterson, S., & Landay, J. A. (2001). WebQuilt: a proxy-based approach to remote web usability testing. *ACM Transactions on Information Systems*, *19*(3), 263–285.

IBM. (2018). Watson Speech to Text. Retrieved February 10, 2018, from ibm.com/watson/services/speech-to-text

Interoperability solutions for public administrations, businesses and citizens. (2018, May). 19th annual international conference on digital government research. Retrieved May 14, 2018, from https://ec.europa.eu/isa2/events/19th-annual-international-conference-digital-government-research_en

Ivory, M. Y. & Hearst, M. A. (2001). The state of the art in automating usability evaluation of user interfaces. *ACM Computing Surveys (CSUR)*, *33*(4), 470–516.

Janssen, M., Charalabidis, Y., & Zuiderwijk, A. (2012). Benefits, adoption barriers and myths of open data and open government. *Information systems management*, *29*(4), 258–268.

Jaro, M. A. (1989). Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida. *Journal of the American Statistical Association*, *84*(406), 414–420.

JavaServer Pages Technology. (2017). Oracle. Retrieved April 25, 2018, from ractive.js.org

JSON. (2018). Introducing JSON. Retrieved July 10, 2018, from json.org

Kauffman, D., Khosmood, F., Kuboi, T., & Dekhtyar, A. (2018). Learning alignments from legislative discourse. In *Proceedings of the 19th annual international conference on digital government research: governance in the data age* (p. 119). ACM.

Kauffman, D., Williams, M., Washington, C., Socher, G., & Khosmood, F. (2018). Multimodal speaker identification in legislative discourse. In *Proceedings of the 19th annual international conference on digital government research: governance in the data age* (p. 66). ACM.

Khosmood, F., Dekhtyar, A., Assai, H., Kurfess, F., & Snyder, J. (2014, August). *Making California legislative process transparent*. California Polytechnic State University, San Luis Obispo.

Knight Foundation. (2015, March). *Assessing civic tech: case studies and resources for tracking outcomes*. Impact Network.

Kohavi, R., Deng, A., Frasca, B., Walker, T., Xu, Y., & Pohlmann, N. (2013). Online controlled experiments at large scale. In *Proceedings of the 19th acm sigkdd international conference on knowledge discovery and data mining* (pp. 1168–1176). ACM.

Kohavi, R. & Longbotham, R. (2017). Online controlled experiments and A/B testing. In *Encyclopedia of machine learning and data mining* (pp. 922–929). Springer.

Lafferty, J. D., McCallum, A., & Pereira, F. C. N. (2001). Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning* (pp. 282–289). ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. Retrieved from http://dl.acm.org/citation.cfm?id=645530.655813

Lam, S. (2016, June). *Digital Democracy Video Manager* (Master's thesis, California Polytechnic State University, San Luis Obispo). Retrieved April 20, 2018, from digitalcommons.calpoly.edu/cscsp/71

Latner, M., Dekhtyar, A. M., Khosmood, F., Angelini, N., & Voorhees, A. (2017). Measuring legislative behavior: an exploration of digitaldemocracy.org. *California Journal of Politics and Policy*, *9*(3).

Legislative Analyst's Office. (2016, November). Proposition 54. Retrieved September 25, 2018, from lao.ca.gov/BallotAnalysis/Proposition?number=54&year=2016

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady* (Vol. 10, *8*, pp. 707–710).

Library of Congress. (2018). Congressional record. Retrieved January 15, 2018, from congress.gov/congressional-record

Luz, S., Masoodian, M., Rogers, B., & Deering, C. (2008). Interface design strategies for computer-assisted speech transcription. In *Proceedings of the 20th australasian conference on computer-human interaction: designing for habitus and habitat* (pp. 203–210). OZCHI '08. Cairns, Australia: ACM. doi:10.1145/1517744.1517812

MapLight. (2017, July). Maplight - revealing money's influence on politics. Retrieved April 25, 2018, from maplight.org

Marshall Plan Foundation. (2017). 2017 scholarship papers. Retrieved September 11, 2018, from marshallplan.at/2017-papers

Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, . . . Xiaoqiang Zheng. (2015). TensorFlow: large-scale machine learning on heterogeneous systems. Software available from tensorflow.org. Retrieved from tensorflow.org

Marzal, A. & Vidal, E. (1993). Computation of normalized edit distance and applications. *IEEE transactions on pattern analysis and machine intelligence*, *15*(9), 926–932.

McCallum, A. & Li, W. (2003). Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the seventh conference on natural language learning at hlt-naacl 2003-volume 4* (pp. 188–191). Association for Computational Linguistics.

Meijer, A. (2015). Government transparency in historical perspective: from the ancient regime to open data in the netherlands. *International Journal of Public Administration*, *38*(3), 189–199. doi:10.1080/01900692.2014.934837

Mozilla and individual contributors. (2018). Ajax - Developer guides. Retrieved July 10, 2018, from developer.mozilla.org/en-US/docs/Web/Guide/AJAX

Mozilla & individual contributors. (2018). MDN web docs - event reference. Retrieved January 10, 2018, from developer.mozilla.org/en-US/docs/Web/Events

Myers, J. (2015, April). Big support in bipartisan poll for a more transparent california legislature. Retrieved April 25, 2018, from ww2.kqed.org/news/2015/04/30/big-support-in-bipartisan-poll-for-a-more-transparent-california-legislature

Nadeau, D. & Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, *30*(1), 3–26.

Needleman, S. B. & Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, *48*(3), 443–453. doi:10.1016/0022-2836(70)90057-4

New Zealand Parliament. (2018). Read Hansard Reports. Retrieved January 15, 2018, from parliament.nz/en/pb/hansard-debates/rhr

Nivre, J., Tullgren, K., Allwood, J., Ahlsén, E., Holm, J., Gronqvist, L., . . . Sofkova, S. (1998). Towards multimodal spoken language corpora: transtool and synctool. In *Proceedings of the workshop on partially automated techniques for transcribing naturally occurring continuous speech* (pp. 11–23). Transcribe '98. Montreal, Quebec, Canada: Association for Computational Linguistics. Retrieved from http://dl.acm.org/citation.cfm?id=1628291.1628293

NLTK Project. (2017a). Natural Language Toolkit - NLTK 3.3 documentation. Retrieved July 20, 2018, from nltk.org

NLTK Project. (2017b). NLTK 3.3 documentation – edit_distance. Retrieved July 25, 2018, from nltk.org/api/nltk.metrics.html#edit_distance

NowTranscribe Ltd. (2017). NowTranscribe audio transcription. Retrieved January 15, 2018, from nowtranscribe.com

Olsson, H. H. & Bosch, J. (2014). From opinions to data-driven software r&d: a multi-case study on how to close the 'open loop' problem. In *Software engineering and advanced applications (SEAA), 2014 40th EUROMICRO conference* (pp. 9–16). IEEE.

Open States. (2017, July). Open states: discover politics in your state. Retrieved April 25, 2018, from openstates.org

Oracle Corporation. (2018). Mysql. Retrieved January 10, 2018, from mysql.com

ORF. (2018). ORF TVthek. Retrieved February 14, 2018, from tvthek.orf.at

Oxford University Press. (2017). Definition of part of speech in English by Oxford Dictionaries. Retrieved April 25, 2018, from en.oxforddictionaries.com/definition/part_of_speech

Paganelli, L. & Paternò, F. (2003). Tools for remote usability evaluation of web applications through browser logs and task models. *Behavior Research Methods, Instruments, & Computers*, *35*(3), 369–378.

Papazian, F., Bossy, R., & Nédellec, C. (2012). Alvisae: a collaborative web text annotation editor for knowledge acquisition. In *Proceedings of the sixth linguistic annotation workshop* (pp. 149–152). LAW VI '12. Jeju, Republic of Korea: Association for Computational Linguistics. Retrieved from http://dl.acm.org/citation.cfm?id=2392747.2392771

Parliament of Australia. (2018). Hansard. Retrieved January 15, 2018, from aph.gov.au/Parliamentary_Business/Hansard

Parliament of Canada. (2018). Publication search. Retrieved January 15, 2018, from ourcommons.ca/parliamentarians/en/publicationsearch

Parliament of the United Kingdom. (2018). Hansard Online. Retrieved January 15, 2018, from hansard.parliament.uk

Parliamentary Office of Science and Technology. (2009). *E-Democracy, POST PN321*. UK Parliament. Retrieved August 11, 2018, from researchbriefings.parliament.uk/ResearchBriefing/Summary/POST-PN-321

Pivotal Software. (2018a, July 26). Core technologies. Retrieved August 1, 2018, from docs.spring.io/spring/docs/current/spring-framework-reference/core.html

Pivotal Software. (2018b, July 26). JPA Repositories - Query methods. Retrieved August 1, 2018, from docs.spring.io/spring/docs/current/spring-framework-reference/core.html

Pivotal Software. (2018c). Spring. Retrieved January 10, 2018, from spring.io

Pivotal Software. (2018d, July 26). Spring AOP APIs. Retrieved August 1, 2018, from docs.spring.io/spring/docs/current/spring-framework-reference/core.html#aop-api

Pivotal Software. (2018e). Understanding REST. Retrieved August 1, 2018, from spring.io/understanding/REST

Pivotal Software. (2018f, July 26). Web on servlet stack. Retrieved August 1, 2018, from docs.spring.io/spring/docs/current/spring-framework-reference/web.html

Public Affairs Office of California Polytechnic State University. (2015, May). Lt. Gov. Newsom and Former State Sen. Blakeslee Trail-blaze Digital Platform to Transform Government Transparency. Retrieved April 25, 2018, from digitalcommons.calpoly.edu/pao_pr/4596

Python Software Foundation. (2018). difflib - Helpers for computing deltas. Retrieved May 26, 2018, from docs.python.org/3/library/difflib.html

Quo Vadis Veritas Redaktions GmbH. (2018). Addendum – Politometer. Retrieved February 14, 2018, from http://tvthek.orf.at

RactiveJS contributors. (2017). Ractive.js. Retrieved January 10, 2018, from ractive.js.org

Ratcliff, J. W. & Metzener, D. E. (1988). Pattern-matching - the gestalt approach. *Dr Dobbs Journal*, *13*(7), 46.

Ratinov, L. & Roth, D. (2009). Design challenges and misconceptions in named entity recognition. In *Proceedings of the thirteenth conference on computational natural language learning* (pp. 147–155). Association for Computational Linguistics.

Red Hat. (2018). Hibernate. Retrieved January 10, 2018, from hibernate.org

Reinman, A. (2016, June). *Improvements to Digital Democracy's Transcription Tool* (Master's thesis, California Polytechnic State University, San Luis Obispo). Retrieved April 20, 2018, from digitalcommons.calpoly.edu/cscsp/96

Revuelta-Martinez, A., Rodriguez, L., & Garcia-Varea, I. (2012). A computer assisted speech transcription system. In *Proceedings of the demonstrations at the 13th conference of the european chapter of the association for computational linguistics* (pp. 41–45). EACL '12. Avignon, France: Association for Computational Linguistics. Retrieved from http://dl.acm.org/citation.cfm?id=2380921.2380930

Robertson, C. (2018, March). *Why Digital Democracy*. California Polytechnic State University, San Luis Obispo. Retrieved March 5, 2018, from drive.google.com/open?id=1lPZOblZ10joM-vUZow8wsmsjk9xn15BT

Rouvier, M., Gay, P., Khoury, E., Merlin, T., & Meignier, S. (2013). An open-source state-of-the-art toolbox for broadcast news diarization. In *Proceedings of interspeech*.

Rovin, J. (2016, March). *Reducing costs in human assisted speech transcription* (Master's thesis, California Polytechnic State University, San Luis Obispo). Retrieved April 20, 2018, from digitalcommons.calpoly.edu/theses/1538

Ruprechter, T., Khosmood, F., Kuboi, T., Dekhtyar, A., & Gütl, C. (2018). Gaining efficiency in human assisted transcription and speech annotation in legislative proceedings. In *Proceedings of the 19th annual international conference on digital government research: governance in the data age* (117:1–117:2). dg.o '18. Delft, The Netherlands: ACM. doi:10.1145/3209281.3209410

Seps, L. (2013). Nanotrans—editor for orthographic and phonetic transcriptions. In *Telecommunications and signal processing (tsp), 2013 36th international conference on* (pp. 479–483). IEEE.

Shin, K., Shafiei, M., Kim, M., Jain, A., & Raghavan, H. (2018). Discovering progression stages in trillion-scale behavior logs. In *Proceedings of the 27th international conference on world wide web (www). acm, forthcoming* (Vol. 1, *1.1*, pp. 1–10).

Smith, T. F. & Waterman, M. S. (1981). Identification of common molecular subsequences. *Journal of molecular biology*, *147*(1), 195–197. doi:10.1016/0022-2836(81)90087-5

Snae, C. (2007). A comparison and analysis of name matching algorithms. *International Journal of Applied Science, Engineering and Technology*, *4*(1), 252–257.

Stanford Natural Language Processing Group. (2018a). Stanford log-linear part-of-speech tagger. Retrieved July 1, 2018, from nlp.stanford.edu/software/tagger.html

Stanford Natural Language Processing Group. (2018b). Stanford named entity recognizer (NER). Retrieved July 1, 2018, from nlp.stanford.edu/software/CRF-NER.html

State of California. (2017a, July). California state assembly website. Retrieved April 25, 2018, from assembly.ca.gov

State of California. (2017b, July). California state legislature website. Retrieved April 25, 2018, from leginfo.legislature.ca.gov

State of California. (2017c, July). California state senate website. Retrieved April 25, 2018, from senate.ca.gov

Stenetorp, P., Pyysalo, S., Topić, G., Ohta, T., Ananiadou, S., & Tsujii, J. (2012). BRAT: a web-based tool for NLP-assisted text annotation. In *Proceedings of the demonstrations at the 13th conference of the european chapter of the association for computational linguistics* (pp. 102–107). EACL '12. Avignon, France: Association for Computational Linguistics. Retrieved from http://dl.acm.org/citation.cfm?id=2380921.2380942

Stripes. (2018). Stripes Wiki - Stripes Framework Wiki. Retrieved February 20, 2018, from stripesframework.org

Surka, M. & Ridlington, E. (2016, April). Following the money 2016. Retrieved April 25, 2018, from calpirg.org/reports/cap/following-money-2016

The Apache Software Foundation. (2018a). Apache OpenNLP. Retrieved July 20, 2018, from opennlp.apache.org

The Apache Software Foundation. (2018b). Apache Tiles. Retrieved April 20, 2018, from foundation.zurb.com

The Apache Software Foundation. (2018c). Apache Tomcat. Retrieved July 10, 2018, from tomcat.apache.org

The Apache Software Foundation. (2018d). Maven. Retrieved August 10, 2018, from maven.apache.org

The Austrian Parliament. (2018a). Erweiterte Suche. Retrieved February 14, 2018, from parlament.gv.at/SUCH

The Austrian Parliament. (2018b). Stenographische Protokolle. Retrieved February 14, 2018, from parlament.gv.at/PAKT/STPROT

The California Channel. (2017, July). The California Channel - politics and public affairs that shape California. Retrieved April 25, 2018, from calchannel.com

The European Parliament. (2018). Debates and videos | planary. Retrieved January 15, 2018, from europarl.europa.eu/plenary/en/debates-video.html

The jQuery Foundation. (2018). jQuery. Retrieved January 10, 2018, from jquery.com

The Swiss Parliament. (2018). Search official bulletin. Retrieved January 15, 2018, from parlament.ch/en/ratsbetrieb/suche-amtliches-bulletin

Tjong Kim Sang, E. F. & De Meulder, F. (2003). Introduction to the CoNLL-2003 shared task: language-independent named entity recognition. In *Proceedings of the seventh conference on natural language learning at hlt-naacl 2003-volume 4* (pp. 142–147). Association for Computational Linguistics.

Toselli, A. H., Vidal, E., & Casacuberta, F. (2011). Computer assisted transcription of text images. In *Multimodal interactive pattern recognition and applications* (pp. 61–98). Springer.

Toutanova, K., Klein, D., Manning, C. D., & Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 conference of the north american chapter of the association for computational linguistics on human language technology-volume 1* (pp. 173–180). Association for Computational Linguistics.

Toutanova, K. & Manning, C. D. (2000). Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the 2000 joint sigdat conference on empirical methods in natural language processing and very large corpora: held in conjunction with the 38th annual meeting of the association for computational linguistics-volume 13* (pp. 63–70). Association for Computational Linguistics.

United States Census Bureau. (2017, July). Annual estimates of the resident population for the united states, regions, states, and puerto rico: april 1, 2010 to july 1, 2017 (nst-est2017-01). Retrieved June 6, 2018, from https://www.2census.

gov/programs-surveys/popest/tables/2010-2017/state/totals/nst-est2017-01.xlsx

Vasconcelos, L. G., Santos, R. D., & Baldochi, L. A. (2016). Exploiting client logs to support the construction of adaptive e-commerce applications. In *e-Business engineering (ICEBE), 2016 IEEE 13th international conference on* (pp. 164–169). IEEE.

Vermeeren, A. P., Law, E. L.-C., Roto, V., Obrist, M., Hoonhout, J., & Väänänen-Vainio-Mattila, K. (2010). User experience evaluation methods: current state and development needs. In *Proceedings of the 6th nordic conference on human-computer interaction: extending boundaries* (pp. 521–530). ACM.

Voit, K. (2018). LaTeX-KOMA-template. Retrieved August 20, 2018, from github.com/novoid/LaTeX-KOMA-template

Watson, G. (2017). OrmLite - Lightweight Object Relational Mapping (ORM) Java Package. Retrieved April 25, 2018, from ormlite.com

Webb, P., Syer, D., Nicoll, J. L. S., Winch, R., Wilkinson, A., Overdijk, M., . . . Bhave, M. (2018, July 26). Spring Boot Reference Guide. Retrieved August 1, 2018, from docs.spring.io/spring-boot/docs/current-SNAPSHOT/reference/htmlsingle

Whittaker, S. & Amento, B. (2004). Semantic speech editing. In *Proceedings of the sigchi conference on human factors in computing systems* (pp. 527–534). CHI '04. Vienna, Austria: ACM. doi:10.1145/985692.985759

Widlöcher, A. & Mathet, Y. (2012). The Glozz platform: a corpus annotation and mining tool. In *Proceedings of the 2012 acm symposium on document engineering* (pp. 171–180). DocEng '12. Paris, France: ACM. doi:10.1145/2361354.2361394

Wijnhoven, F., Ehrenhard, M., & Kuhn, J. (2015). Open government objectives and participation motivations. *Government information quarterly*, *32*(1), 30–42.

Winkler, W. E. (1990). String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. In *Proceedings of the section on survey research*.

Yancey, W. E. (2005). *Evaluating string comparator performance for record linkage*. Bureau of the Census.

ZURB, Inc. (2018). Foundation. Retrieved August 20, 2018, from foundation.zurb.com