Retention of User Edits in Content Analysis Output Combination and its Utilisation in Digital Repositories

A thesis

submitted in fulfilment

of the requirements for the Degree

of

Master of Science in Computer Science

at

The University of Waikato

by

Finn Innes

THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

2023

Abstract

This thesis investigates how repeated applications of content analysis algorithms applied to a document corpus can be incorporated with human corrections in such a way that the corrections are not erased should the corpus be re-processed. Diarisation (who speaks when) is the focal content analysis type, however the edit-aware concepts apply to other areas such as OCR and Automatic Speech Recognition (ASR). As a means for experimentation, an enriched audio player is built to provide the ability to visualise diarisation data. This web tool is further developed to allow for the editing of speaker region bounds and speaker names, the locking of corrected regions, and the comparison between two diarisation outputs. Sequence merging and region conflict concepts are also explored and implemented. The audio tool is integrated into the Greenstone Digital Library Software, which is used as an example digital library framework. The concepts explored are outlined in a way that allows the strategies to be transferable to alternative content analysis systems or digital libraries.

Acknowledgements

I would like to express my gratitude to my supervisor, Dr. David Bainbridge, for his guidance and invaluable mentorship throughout the course of my master's thesis. His expertise and assistance have been instrumental in shaping my research and academic pursuit.

Additionally, I would like to thank Tiriana Anderson, for helping develop ideas and providing discussions.

My heartfelt thanks extend to my partner Jñāna, my sister Tessa, and parents John and Gabi, whose encouragement, support, and camaraderie provided a source of motivation throughout the duration of my thesis.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Notwithstanding the Information Age in which we live, a significant challenge faced as a society is how the vast array of physical content—that still exists and continues to be produced—can be brought into the digital realm. The availability of digitisation tools has brought with it an explosion of electronic content whether it is text, audio, images, or other multimedia formats, but in an impoverished form when access to the actual content it represents is considered.

Beyond the actual digitisation, to develop digital resources based on physical items such as a collection of documents or interview recordings on audio cassettes, a common method is to apply content-based analysis such as Optical Character Recognition (OCR) or Automatic Speech Recognition (ASR) to enrich access to the digital representation of the resource. Such algorithmic forms of processing, however, are prone to errors and rarely produce perfect results. Therefore, in situations where a fully accurate presentation is required, manual correction is undertaken to complete the process.

This latter step is generally time-consuming to achieve as it requires human effort. At the heart of such an undertaking, a tension exists due to the difference in the time it takes for content analysis, which is extremely fast, in comparison to the process of manual correction. Layered on top of this, the manual correction process might be organised as a multi-phased work-

flow, for example ensuring high-value text such as the title and authors of the documents in the corpus are corrected first before other aspects are addressed. Therefore, a critical issue arises when an improved version or better alternative of the content analysis algorithm becomes available and the full undertaking of manual corrections has not been completed—if the new version is used to process the corpus, what happens to the manual corrections? The interaction of how the automated forms of content analysis can operate in unison with manual corrections is the focus of this work.

To assess the impact of the rise of digital forms of information on an otherwise predominantly analogue world, a study exists which involved a wide-ranging investigation of how we compute, communicate and store information, spanning the years 1986–2007 (Hilbert and López, 2011). Concerning the storage of information, their analysis found this to be increasing at a rate of 23% per year. While this work was undertaken over a decade ago, a detail of note in their work was the ongoing trajectory of how much information in analogue form continues to be produced. The transformation of analogue content to digital form opens up new avenues of access, use, and research, delivering information-rich material to an increasingly large audience (Tanner, 2001).

Trove, as an example, is a digital repository produced by Australia's community, cultural, and research institutions, spearheaded by the National Library of Australia (Holley, 2010). They provide a publicly accessible website that hosts an extensive selection of historical content, some of which has been scanned and processed through an OCR system, offering full-text searching, as shown in Figure 1.1.

Relevant to the focus of the thesis, Trove allows users to edit the OCR results in place when viewing a document, as illustrated in Figure 1.2. Similar to Wikipedia, Trove is self-moderated, putting trust in the community to follow editing guidelines. Text can be edited multiple times, ideally improving with each new version. Trove employs a multi-step ingest process for incorporating content into the digital library, which sees them collaborate with multiple

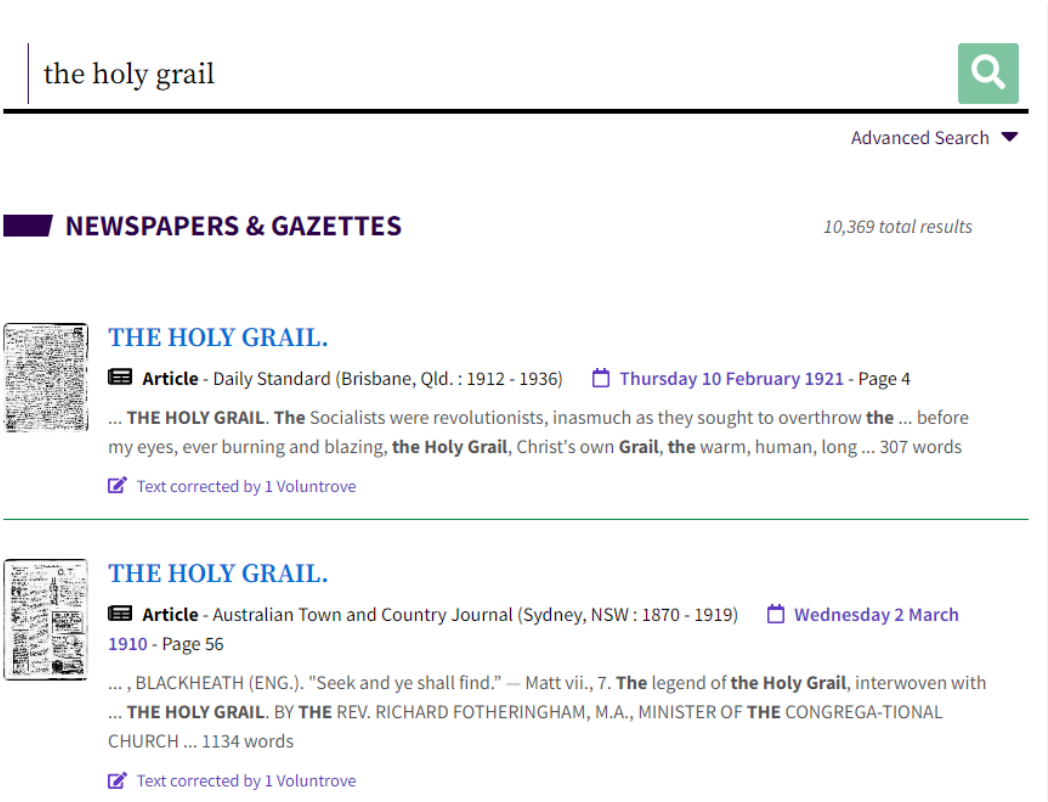

Figure 1.1: Trove Searching

OCR specialists.[1] While the workflow utilised in Trove involves a feedback loop through which improvement in the final output is possible, this is between the OCR contractors and the quality assurance team in Trove. Manual corrections only occur once the item is published in Trove. The devised workflow, therefore, can accommodate improvements in the OCR tools used for new content; however, not so for existing content.

Another repository that utilises content analysis to enrich its digital content is the Duke University Libraries (Daniel et al., 2011). Their online archive provides access to scholarly collections, items such as newspapers and photographs, video, audio, and other forms of digital content. Several audio collections also provide interactive transcriptions, allowing the listener to see the result of a speech-to-text process.[2] Figure 1.3 shows an audio item in a digitised collection, with the transcribed text underneath the audio player. Clicking on

[1] https://trove.nla.gov.au/partners/partner-services/digitise/digitisation-process

[2] https://blogs.library.duke.edu/bitstreams/2017/10/24/accessible-av-duke-digital-repository/

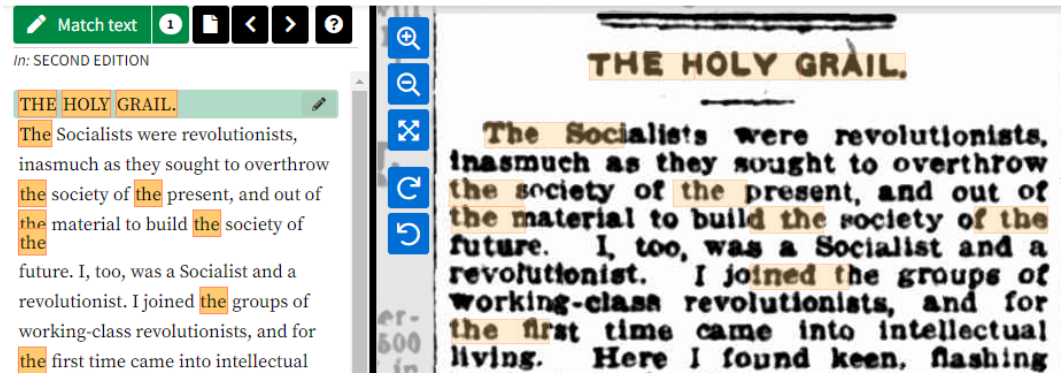

Figure 1.2: Trove OCR User Corrections (*Note:* Users are required to create a free account in order to edit OCR text. This is the first layer of protection against users with malicious intent.)

a transcribed line advances the player to the corresponding moment in the media. The use of transcriptions in the library outside of the user interactions is unknown, however, the implications of extracting further information from the transcribed text go beyond access to the full-text. Such processes could include keyword, acronym, language, and sentiment analysis, providing numerous avenues to enrich the digital content.

Significant advances have been made in the field of Artificial Intelligence (AI) through the development of deep learning models (Došilović et al., 2018; Makridakis, 2017). A prime example of this is the OCR capability in Google Vision, which can recognise handwritten text in addition to printed form in over 200 languages (Walker et al., 2018). Such tools have enabled substantial progress in computer vision tasks, making them a cornerstone in the development of OCR systems (Singh et al., 2012).

Another example is Microsoft's Azure speech-to-text capability, which uses deep learning models to convert audio streams into text, supporting over 250 languages and offering real-time transcription (Salvaris et al., 2018). These sorts of advances are exactly the circumstance that would make it beneficial to reprocess the content of digital repositories. However, as the Trove example demonstrates—which is considered state-of-the-art concerning image-based digital library solutions for legacy content—the intricacies involved to achieve

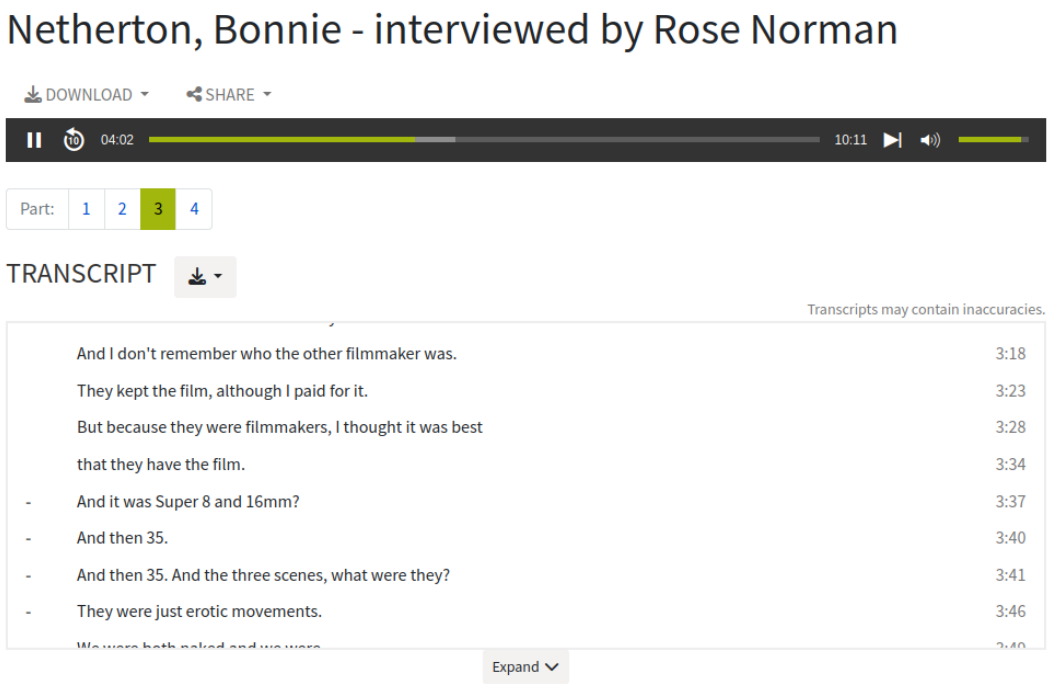

Figure 1.3: Duke University Libraries Audio Transcript View

this go beyond what can be developed using standard software components, even when a bespoke digital library solution has been developed.

1.1 Objectives

The primary objective of this thesis is to develop a method for retaining user edits in an online environment where data generated from automated content analysis can be run and re-run. More specifically, the aim is to generate an approach that allows arbitrary interleaving of automated content analysis and manual user correction of the same digitised document.

The work reported in the thesis was undertaken in the context of a digital library, a common form of online resource where digital content is accessed and managed. Typically, documents in a digital library undergo enrichment of the digital form when it is ingested into the digital library. This enrichment process dovetails with other digital library specific functions such as the addition of metadata to documents. Information extraction and metadata addition both work towards enriching documents within a digital library, improving item

discoverability. Content destined for a digital library that starts life in the physical form needs to be digitised as part of this process. Advanced forms of digital library architecture make use of OCR and ASR as part of the ingest process, making this form of digital repository a suitable environment in which to investigate the goal of the thesis.

1.2 Contributions

The main contributions of the thesis are:

- A review of existing digital library systems, including an appraisal of their support for diarisation methods and other content analysis techniques to identify and categorise their abilities to support and (ideally) retain user edits.
- The implementation and extension of an existing algorithm which merges multiple streams of content analysis outputs. The added functionality allows for sections of the input streams to be locked, ensuring such sections remain present in the final output, thereby providing a mechanism by which manual corrections can be protected.
- The extension of a web tool—which allows audio to be played and speaker segments to be visualised within a digital library context—to include the ability to correct diarisation errors and rename speaker segments. The capability of the editor is further enhanced by providing the ability to run the aforementioned extended algorithm for merging multiple streams of content analysis output.
- A discussion of the findings and implications of the research, highlighting the contributions made to the field of diarisation and content analysis, and proposing potential future enhancements.

The methods explored will not pertain only to the environment it is developed in but apply to other environments and strategies. By addressing the aforementioned focal points, this project aims to contribute to the field of

diarisation and provide a capable audio analysis tool in a digital library.

1.3 Synopsis

The remainder of the thesis is structured as follows. In Chapter 2, digital libraries and content analysis techniques are explored. This involves an analysis of digital library components and a survey of existing systems, and an overview of content analysis systems and how they can be integrated into digital libraries. This introduces concepts that will be developed further in following chapters.

Chapter 3 conducts a review of content analysis improvement approaches, including machine learning, human involvement, and output combination strategies. The benefits of each approach are discussed, and drawbacks highlighted to accentuate areas where gains are possible when viewed in the context of user corrections within digital libraries.

In Chapter 4, the algorithm for the lock-aware diarisation combination technique is presented. Chapter 5 describes the developed diarisation editor tool. The libraries used are examined to illustrate the foundation upon which the tool is built. The architecture and user interface are then described with supporting visual resources. To tie in the previous chapter's developments, the interactions with the newly created script are outlined.

The thesis concludes in Chapter 6, where the key findings and concepts are accentuated, and contributions are reiterated. Potential future work is also outlined, setting the scope for how the completed work could be further developed.

Chapter 2

Background

This chapter starts by exploring digital libraries, their components, and outlines a set of digital libraries relevant to the area of study that support user-based editing. This sets the stage by showing what bespoke digital library toolkits provide in terms of content management and correction. By examining digital libraries and their supported content analysis integrations, relevant concepts are introduced which are later expanded and built upon in the following chapters.

As well as exploring digital libraries, more general forms of output combination systems are explored. These systems take multiple outputs from one or more content analysis systems and derive a consensus output that aims to reduce the overall error rate. Both OCR and diarisation output combination tools are examined, as well as notable ASR combination tools. This thesis proposes the integration of such a concept within a digital library to improve the data associated with the documents within such a repository.

As the examples given in Chapter 1 show, the field of content analysis within digital libraries has witnessed significant advancements in recent years, contributing to improved information extraction and understanding. This section provides a comprehensive background review of the primary objectives of this thesis. The initial part of this literature review delves into the existing digital libraries along with content analysis techniques. Subjects such as

crowdsourcing, citizen science, and document editing within digital libraries are explored.

2.1 Digital Libraries

Unlike a bricks-and-mortar library, a digital library has no physical limits to information storage. Generally, documents can be accessed at any time by multiple users, providing convenient and efficient information retrieval globally. A widely used definition for a digital library is a computer system that provides "a focused collection of digital objects, including text, video, and audio, along with methods for access and retrieval, and for selection, organization and maintenance" (Witten et al., 2009).

Digital library systems fall into two broad categories: custom-built and general-purpose. The latter examples are alternatively referred to as toolkits. Trove, mentioned in Chapter 1, IEEE Xplore Digital Library (Durniak, 2000), and PubMed (White, 2020) are examples of custom-built digital library systems.

General-purpose digital library toolkits, in comparison, allow an individual or organisation to build their own collection of content, manage access, and customise the user experience. Extending the comparison of a library with a digital library, a user of the latter who engages in the management of the content is often referred to as a digital librarian, and a category or user very much relevant to the work reported in the thesis.

Whether utilising a custom-built digital library or using a general-purpose toolkit, an ingestion process is used to form the digital library using an assembly of given documents. Document ingestion varies between different digital library systems. Some require documents to be in a standardised format, whereas others automatically convert documents in a range of formats into a more suitable format for processing by the digital library software. Most digital library systems will extract full-text from the ingested documents, however,

there are variations between digital library systems over concepts that might be systematically determined, such as the document title or language the text is written in.

2.1.1 Full-Text Indexing, Search, and Metadata

Most digital libraries provide some form of search interface which helps users find desired resources. As the volume and diversity of digital library content expands, the effectiveness of the search mechanism becomes a crucial component of the user experience.

Full-text indexing is the process of creating the data structures necessary to efficiently specify every single document that contains the terms contained in a given query, mapped to the file system for reasons of persistency (Grainger and Potter, 2014; Witten et al., 1999).

Beyond this core requirement, full-text indexing tools may (or may not) support case sensitive/insensitive matching, accent folding, and stemming, in regard to how query matching can be performed, amongst other features. Representational choices in the data structures used can also impact whether new documents can be added incrementally, whether searching can be at a finer level of granularity, if stop words are omitted or not, and whether sufficient details are stored in the index to reconstruct the original document.

Metadata extraction is a vital step that can enhance the searchable content in a library. The Dublin Core (DC) Metadata Registry is the most commonly used metadata schema for web content (Weibel et al., 1998). It is supported, for example, by all of the digital library systems reviewed in this chapter. DC defines a set of fifteen metadata elements containing items falling within content, intellectual property, and instantiation categories. A digital library may offer further schemas or customisable metadata entries to enrich the information attached to a document. In addition to search, metadata plays an important role in a digital library in forming structures that allow users to browse the content store. Such structures may include subject keywords or

titles sorted alphabetically.

In a digital library, when the starting document format is image, audio, or video, content analysis tools can be used to generate textual information that can be incorporated into the digital library as metadata. This enriches the forms of access available for locating the document. OCR is an example of turning a non-textual type of media (image) into machine-readable, therefore indexable, text. ASR and diarisation may be applied to audio documents to extract transcription content and speaker turn information. Other media holds searchable data such as speaker or chapter names in audio, object or face detection in images, and themes or actors in video. Metadata may be automatically extracted when it is added to a digital library or added manually by a librarian.

2.1.2 Exemplar Digital Libraries

The following is a review of existing digital library toolkits. As published papers about these digital library systems do not often go into the level of detail needed to assess their level of support for the capabilities sought, as part of this survey, each toolkit was installed and tested to determine relevant information pertinent to the features under investigation. Regarding installation, the majority of toolkits involve the setup of a web server, initialisation of a database, and a collection configuration process.

Omeka

Omeka Classic[1] is an open-source web publishing platform for sharing digital collections and creating media-rich online exhibits (Cohen, 2008). Omeka also produces "Omeka S" which is built for institutions managing a sharable resource pool across multiple sites. Omeka Classic 3, the latest version of the Classic series at the time of writing, is more tailored towards individual projects and educators. There are over 100 plugins available developed by both

[1] https://omeka.org/classic/

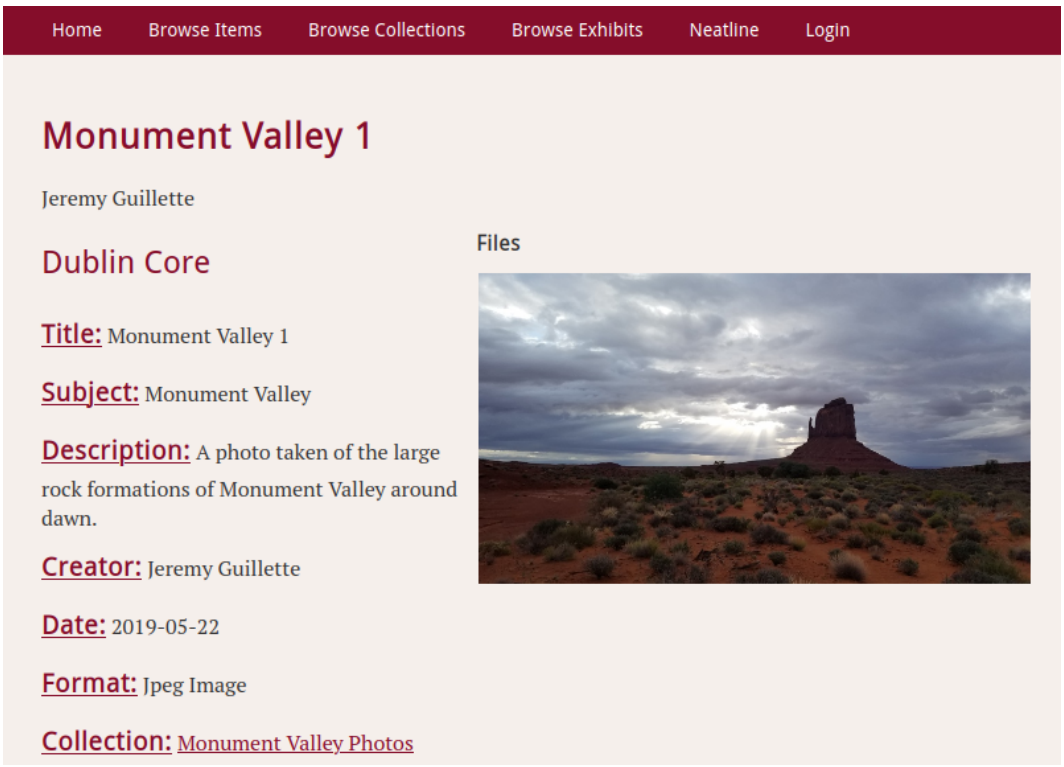

Figure 2.1: Omeka 9 Demonstration Site (*Note:* The demonstration website is provided by Harvard University to showcase Omeka's capabilities.)

users and Omeka which enrich Omeka's capabilities and user interactions, such as a *Commenting* plugin[2] which allows for comments to be added to digital documents, and the *YouTube Import*[3] plugin, giving the ability for YouTube videos to be imported and displayed in Omeka.

Omeka has proved useful in crowdsourcing projects such as *The Louisville Leader* transcription project, where Omeka was used to host newspaper documents that had undergone OCR (Daniels et al., 2014). A third-party module, Scripto, was employed to allow users to edit the OCR text, to correct errors.

Omeka requires a web server and an instance of MySQL to operate. Omeka Classic is free and open-source and also offers various tiers of hosting for those wanting an easy startup. Figure 2.1 shows a demonstration website[4] showcasing Omeka's capabilities hosting a collection of images with attached metadata.

[2]https://omeka.org/classic/plugins/Commenting/
[3]https://omeka.org/classic/plugins/YouTubeImport/
[4]https://demo.omeka.fas.harvard.edu/

Fedora

Fedora[5] is a standards-based, open-source repository designed for the robust preservation of digital content (Staples et al., 2003). Fedora is under the stewardship of Lyrasis, a non-profit organisation serving and supporting libraries, archives, and cultural heritage organisations globally. Fedora offers a REST interface allowing users to interact with collection items and import/export resources.

Fedora version 6 was released in 2021, bringing a simple search interface and performance improvements to the toolkit. For installation, Java 11 and a web server are required. Fedora's document model has evolved over time, now sporting an architecture that includes a Resource Description Framework (RDF)-based object model to represent relationships between objects and their components (Lagoze et al., 2006).

Individual documents can be created through the web interface. Large-scale ingest focuses on documents being in a standardised form, FedoraMETS—a Fedora-specific extension of Metadata Encoding and Transmission Standard (METS) (Cundiff, 2004). Fedora provides a RESTful API which exposes functions allowing for more bespoke ingest workflows and custom integrations.

Drupal

Drupal[6] offers robust content management tools and sophisticated APIs for multichannel publishing (Corlosquet et al., 2009). Drupal brings with it a collection of over 50,000 community-maintained modules to customise a Drupal installation. Drupal module functionalities range from improving the user interface to OCR integration. While Drupal is not typically classified as a digital library solution in its default configuration, there are a variety of digital library-based extensions such as Islandora and Mukurtu (described below) that

[5]https://fedora.lyrasis.org/

[6]https://drupal.org/about/

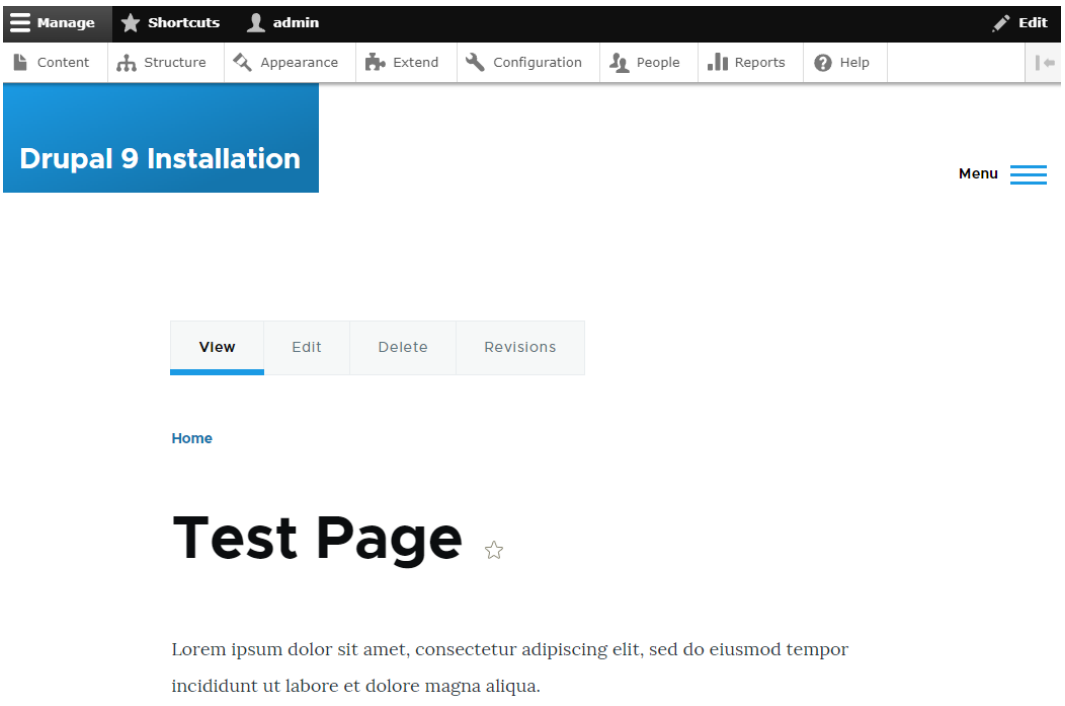

Figure 2.2: Drupal 9 Test Installation

expand the set of features it provides.

A web server is required as well as a relational database for installation. Figure 2.2 shows the administrator's view of a Drupal 9 installation. By default, only two types of content can be created: *Article* and *Basic Page*. Files and images may then be attached to these pages. From the dashboard, an administrator can control the content, collection configuration, enabled plugins, and access status reports and log messages.

Islandora

Islandora[7] is an open-source framework that provides the necessary tools to use a Drupal website as a fully functional digital assets management system (Leggott, 2009). Islandora is built upon Drupal and uses Apache Solr as the engine for full-text indexing and querying. Additionally, Islandora can integrate Drupal modules, allowing for extensive expansion.

Similar to Drupal, both an *Article* and *Basic Page* item can be created in the digital collection. Additionally, Islandora 8 allows a *Repository Item*

[7]https://islandora.ca

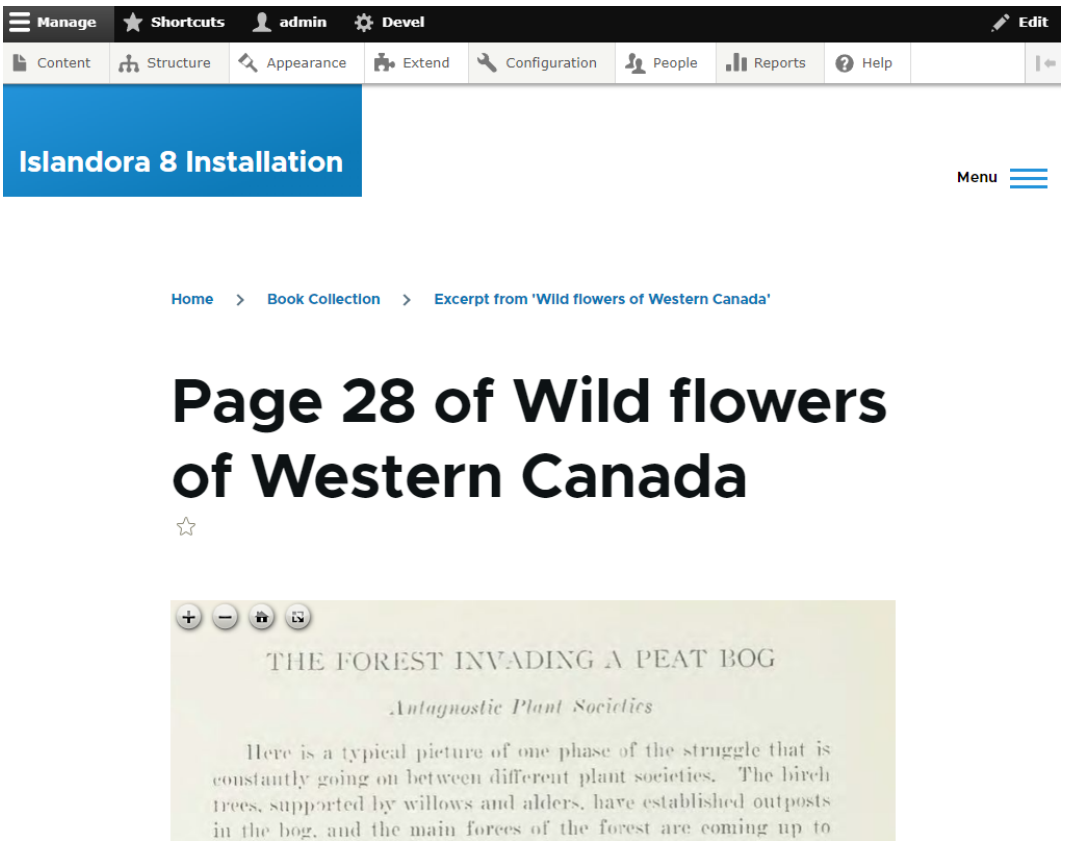

Figure 2.3: Islandora 8 Demo Installation

to be created, representing a type of media (Audio, Image, Newspaper, or Video) or another digitally formed item. In comparison with Drupal, creating a digital repository in Islandora is substantially more suitable due to this capability. The same administrative options are available as Drupal, with extended options in most menus. In Figure 2.3, the similarities in the user interface between Drupal and Islandora can be noticed.

Although OCR is not a core functionality provided by Islandora, a custom module exists that allows for OCR integration using Tesseract. As with Omeka, Islandora can be extended to use the Scripto Module, allowing transcribed text to be edited by users. This module will be outlined in a subsequent section.

Mukurtu

Mukurtu[8] is a free, mobile, and open-source platform developed with Indigenous communities to manage and share digital cultural heritage (K. A. Christen, 2012). Built upon Drupal, Mukurtu puts particular attention towards cultural protocols. An example of this is integrating Traditional Knowledge (TK) Labels (K. Christen, 2015). TK labels are a set of digital tags used to protect and recognise the contributions of indigenous communities by granting them rights and control over their knowledge and data (Montenegro, 2019).

Mukurtu requires a web server and a relational database. Mukurtu supports audio, video, images, and other common document types. Figure 2.4 shows a demonstration website created using Murkutu, accessible through Murkutu's website. The TK Labels can be noticed in the side panel, denoting cultural handling rights surrounding the selected document.

CiteSeerX

CiteSeerX[9] is a public search engine and digital library for scientific and academic papers, primarily in the fields of computer and information science (Wu et al., 2015). CiteSeerX has the capacity to accept user corrections on metadata attached to documents within a library. There are no avenues available for expansion via extensions or modules, however, it integrates numerous extraction algorithms to detect headers, authors, and citations for accurate metadata attachment, which enrich the document representation and consequently, the search process.

While CiteSeerX is an open-source project, developer effort in this project centres around the publicly available installation. As part of this review, it was not installed locally as it can primarily be viewed as an example of a custom-built digital library instead of a digital library toolkit. It has been included in this survey as it demonstrates the integration of automatic algorithms at doc-

[8] https://mukurtu.org/

[9] https://citeseerx.ist.psu.edu/

Figure 2.4: Mukurtu Demo Site (*Note:* TK labels were designed to be utilised by communities who, due to colonial practices of collecting and Western definitions of authorship and ownership, are unable to assert legal control over their collections (Montenegro, 2019).)

ument ingest, which is a desirable feature and relevant to information retrieval in digital libraries.

Greenstone

Greenstone[10] is a suite of software for building and distributing digital library collections. Greenstone is expandable via its plugin infrastructure, allowing a variety of input documents to be imported.

Greenstone 3 is a complete reimplementation of Greenstone 2, offering interface customisation, active updates, and supporting multiple sets of collections in a single installation. Greenstone 3 is backwards-compatible with Greenstone 2, making migration to the newer version straightforward.

[10]https://greenstone.org/

Figure 2.5: Greenstone 3 Demonstration Installation

A web server and Java support are required for Greenstone 3 to be installed. Figure 2.5 displays a Greenstone 3 installation with a demonstration document shown. To initialise a collection, the *Greenstone Librarian Interface* (GLI) may be used. Once documents have been imported and formatting statements adjusted to the digital librarian's liking, the collection can be activated and viewed in a web browser.

In its default configuration, Greenstone 3's document enrichment concentrates on text-based structural details that can be determined from processing the documents. Such details include title, author, and keyword assignment, as well as the determination of section headings and pages. Through its extension mechanism, content analysis techniques may also be applied. Options exist for using both a local instance of Tesseract and an automated process using the Google Cloud Vision API, which can perform both OCR and object classification. Such extracted content is incorporated into the digital library collection as full-text and metadata to enhance the search results.

Veridian

Veridian[11] aids in preserving archives of historic material and delivers the content as a digital collection to their communities. Built upon Greenstone

[11] https://veridiansoftware.com/

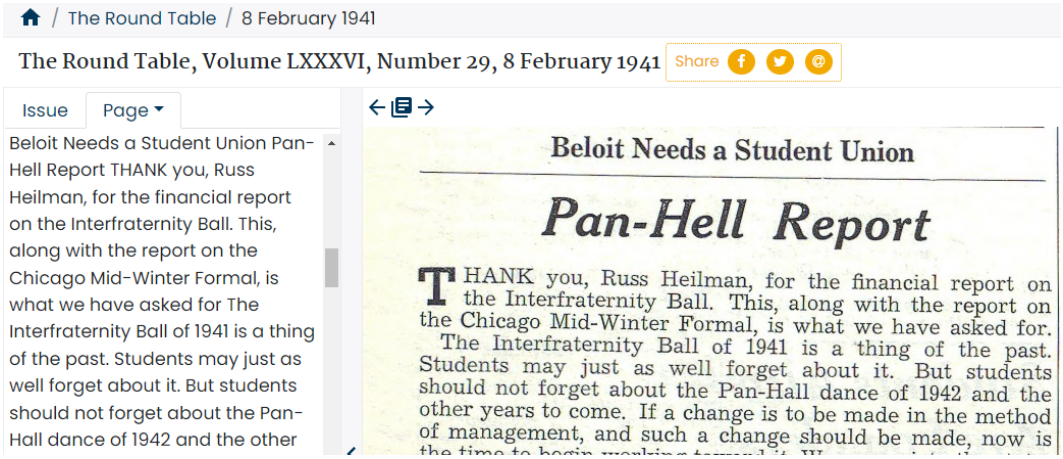

Figure 2.6: Veridian Example Project

2, Veridian offers tailor-made paid solutions for creating digital library collections. Collection digitisation and conversion processes are available for a premium cost. Veridian promotes a particular focus on archival preservation and cultural heritage.

Figure 2.6 shows a demonstration of a Veridian digital library with OCR text on the left and an image of the newspaper on the right. This particular collection of the Beloit College Publications Archive contains articles, advertisements, and photographs of various publications including content from 1919 to 2004[12].

2.1.3 Discussion

While similarities can be observed between the surveyed DL toolkits, contrasts are noticed when investigating features available to the user. For example, the CiteSeerX toolkit is primarily a custom-built digital library with its source code made publicly available. This contrasts with Omeka, which even offers its own hosting solutions and a more digital-librarian-oriented approach. Whereas the majority of toolkits require enriched content to be provided from an external source, Greenstone is a notable exception, where it integrates content analysis processes as the input documents are ingested. Regarding these systems' abil-

[12]https://digitalpublications.beloit.edu/

ities to support content analysis along with the manual correction of errors, a comparison of their relevant features is explored in the next section.

2.2 Content Analysis

Content analysis involves systematically analysing textual, visual, or audial data to extract useful insights and information (Harwood and Garry, 2003). Production-ready content analysis systems should have the ability to handle large amounts of data, provide replicable results, and uncover latent content in the data. Examples of content analysis include language detection and keyword extraction in text, OCR and object classification in images, transcription and diarisation in audio, as well as scene detection and object classification in video.

The quality of data used as input into content analysis systems has a strong correlation to the quality of output. If a scan is grainy or audio contains significant background noise, the content analysis system will struggle to extract data with high accuracy.

2.2.1 Content Analysis in Digital Libraries

Content analysis is an efficient method of expanding the available information attached to digital documents. Content analysis capabilities in digital libraries fall into two categories: external and internal. External content analysis tools are those that are not developed and integrated by the DL toolkit itself. Instead, community developers or separate companies create such tools and make them available through the DL toolkit's plugin or extension list. Internal content analysis tools are developed and integrated by the DL toolkit's developers and generally reflect the importance of information extraction to the particular software.

OCR is a valuable tool to be made available in DL systems as it extracts text from scanned documents or newspapers which can then be indexed and

searched for. OCR providers such as Google Cloud Vision and Tesseract have made implementing OCR straightforward and computationally cheap. While today's OCR tools are capable of impressive accuracies, errors still exist and are more prolific if the imagery is of poor quality. To rectify OCR errors, digital libraries can allow the user to manually edit the extracted text, correcting any mistakes.

To support textual corrections, version history is another key feature that allows changes to be saved and viewed. This is important as it allows document versions to be reverted if poor edits are made. Table 2.1 lists a collection of tested digital library software and the content analysis systems implemented in each. Additionally, manual text transcription availability and version history support are also given. Version history is grouped in as it is an integral element to user corrections within digital libraries. Without the capacity to access previous versions, the original document could be lost.

Automatic extraction of text (OCR) and spoken words (transcription) in video (Sato et al., 1999), keyword and sentiment extraction in textual documents, and transcription or diarisation in audio all enable and/or improve the searchability of an item in a digital library. The ability to find an item in a digital library is a critical part of the digital library environment. Content analysis allows for traditionally non-indexable items to be indexed and enriches the search depth of traditionally indexable items (text).

Table 2.2 outlines the target audience, expandability, and whether or not the target software is open source. The capability to integrate modules or plugins developed by the digital library developers or by the community allows for a plethora of features that otherwise would be inaccessible. For example, the Greenstone, Omeka, and Drupal-based digital libraries offer plugins that expand the variety of importable file types.

	Content Analysis Features	**Manual Transcription**	**Version History**
DSpace	External: OCR by 4Science	External: 4Science	No
Omeka Classic 3	No	Yes (Scripto and other modules)	Yes (Via Module)
Fedora 6	No	No	Yes
Drupal 9	External: OCR, sentiment, keyword	Yes (Scripto)	Yes
Islandora 8	OCR by Tesseract + External: Sentiment, keyword	Yes (Scripto)	Yes
Mukurtu 3	External: OCR, sentiment, keyword	Yes (natively and Scripto)	Yes
CiteSeerX	Author disambiguation, metadata extraction	Yes (metadata correction)	Yes (staff only)
Veridian	OCR	Yes (User Text Correction)	Yes
Greenstone	OCR	Yes (metadata correction)	No

Table 2.1: Digital Library Features

	Target Audience	**Expandable via Modules**	**Open Source**
DSpace	Universities, Non-profit and Commercial Operations	Yes (3 + Various External Sources)	Yes (BSD)
Omeka Classic 3	Individual Projects and Educators	Yes (over 100)	Yes (GNU GPL)
Fedora 6	Universities and National Agencies	No	Yes (Apache)
Drupal 9	Universities, Media Companies, and National Agencies	Yes (over 50,000)	Yes (GNU GPL)
Islandora 8	Universities, Libraries, and Archives	Yes (Drupal Modules)	Yes (GNU GPL)
Mukurtu 3	Indigenous Communities	Yes (Drupal Modules)	Yes (GNU GPL)
CiteSeerX	Scientific Literature Insitutions	No	Yes (Apache)
Veridian	Cultural Heritage Insitutions and Libraries	Yes (Tailor Made)	No
Greenstone	Universities, Libraries, and Public Service Institutions	Yes (over 50)	Yes (GNU GPL)

Table 2.2: Digital Library Features continued

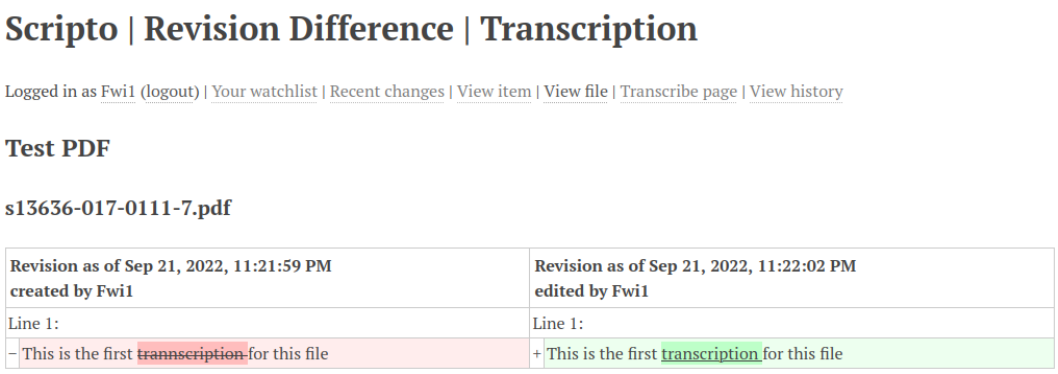

Figure 2.7: Scripto Plugin in Omeka Installation (*Note:* A screenshot of the Scripto plugin installed in an Omeka digital library. Scripto's "diff" ability is showcased, which allows the comparison between versions of transcriptions.)

Scripto

Made available in Omeka, Drupal, Islandora, Mukurtu, and other exemplar digital library toolkits, Scripto is a tool that allows users to transcribe digital files in an online collection. Scripto encourages manual transcription, translation, or descriptions for digital files, improving the attached information and discoverability. The tool is built upon MediaWiki which is the software powering Wikipedia. When a user submits an edit to a page, MediaWiki writes it to the database, without deleting the previous versions of the page, thus allowing easy reversion in cases of vandalism or spamming (MediaWiki, 2023).

Figure 2.7 shows the transcribed text of a document in Omeka after a spelling error revision. The added feature of visualising differences between edit versions helps conceptualise changes.

2.2.2 Drawbacks

A drawback of content analysis systems is that they generally fall short of 100% accuracy (Holley, 2009; McCrocklin et al., 2019). Additionally, the accuracy of OCR, diarisation, and transcription systems relies heavily on the quality of input. A remedy for improving the accuracy rates involves preprocessing or postprocessing. In the case of audio transcription, reducing noise is an example

of a preprocessing measure and may help the system produce more accurate results. Conversely, using vocabulary adjustments such as grammar and N-Gram models is an example of using postprocessing to improve the analysis results.

2.3 Greenstone Digital Library

Greenstone "is not a digital library per se, but a tool that empowers others to build and distribute fully-searchable, metadata-driven digital collections of their own content, accessed from the web or removable media such as a USB thumbdrive or DVD" (Bainbridge and Witten, 2020). Greenstone is an open-source suite of software produced by the New Zealand Digital Library Project at the University of Waikato issued under the General Public License (GNU). Due to its relationship with the University of Waikato and capability to execute content analysis at ingest, it will be the digital library software architecture in which the technical work for this thesis is undertaken.

Document processing plugins are incorporated into Greenstone to handle the input of various file types. They are also a method of generating extra metadata or converting between file types to retrieve and showcase more relevant information within the library. Greenstone is a metadata-based digital library toolkit, meaning a particular emphasis is observed on extracting, storing, and using metadata from digital documents.

Full-text indexing is a valuable feature present in numerous digital libraries. It is a method of organising textual data to enable efficient searching and retrieval of information. Full-text indexing is commonly used in search engines and databases to improve the speed and accuracy of search results. Greenstone gives digital librarians the choice of using either MG, MGPP, Apache Lucene, or Apache Solr as the collection's search indexer.

Incremental indexing is the process of adding new or modified data to an existing index without re-indexing the entire dataset, which helps improve

efficiency and speed when updating information. Both Lucene and Solr can achieve incremental indexing, whereas MG and MGPP cannot. In comparison, MG and MGPP use compression in their full-text indexes with the added capability of being able to reconstruct the original text of a document from the index, meaning, unlike Lucene and Solr, a copy of the document does not need to be retained.

Greenstone 3 is a reimplementation of the original Greenstone 2. The newer edition makes more substantial use of standard web component technologies and sports features such as user interface themes, sets of multiple collections in a single installation, and a RESTful API.

2.3.1 Document and Metadata Ingest

The Greenstone collection build process consists of three stages: import, build, and activate. It is possible to execute each of these stages separately using the following commands: *import.pl*, *buildcol.pl*, and *activate.pl* along with parameters such as the collection name. Figure 2.8 illustrates the ingest pipeline, beginning with the input documents and generating the files necessary to serve the collection. The entire process begins with files being assembled by the digital librarian. Here we concentrate on the operational mechanics of Greenstone through its command-line scripts and the files these scripts generate, as this underpins the work detailed in Chapter 5. In reality, for a digital librarian who is creating and managing a digital library, over time, these details are abstracted away through the GUI-based Greenstone Library Interface.

Import

Importing a collection is the process of converting the assembled input documents into a standardised XML-based format. This is also where metadata is associated with the documents. Manually assigned metadata may be supplied in a file separate from the document being ingested (typically named *metadata.xml*), or it may be extracted from the file, or both.

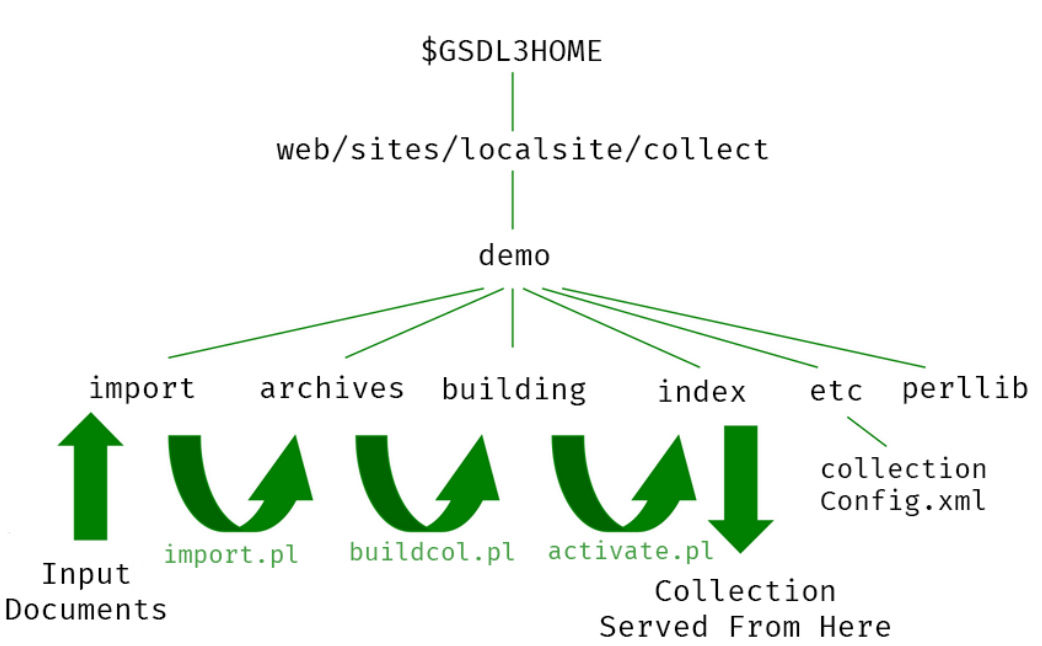

Figure 2.8: Greenstone 3 Building Process

Document processing plugins, conceptualised as a pipeline down which documents are sent, are central to the import process. Plugins are written in Perl and convert documents and metadata into a uniform XML representation that is then used throughout the system (Witten et al., 2002). Figure 2.9 shows a Greenstone plugin pipeline, where files are accepted by their corresponding plugin which extracts metadata and performs processes specific to the file type. Here, "corresponding plugin" generally is based on the file extension, although the designed architecture is flexible enough to allow plugins to open up and scan the content of a file, if needed, to determine if it is a file that is able to be processed.

Inheritance is used in the design of the Perl plugins, with a base class provided that can perform common processing tasks. Such tasks include assigning metadata representing the detected language and the character encoding of the text. Additionally, enrichment tasks are available such as the addition of metadata denoting detected key phrases and acronyms.

In the given example, the *EXCEL* plugin provides a method of attaching

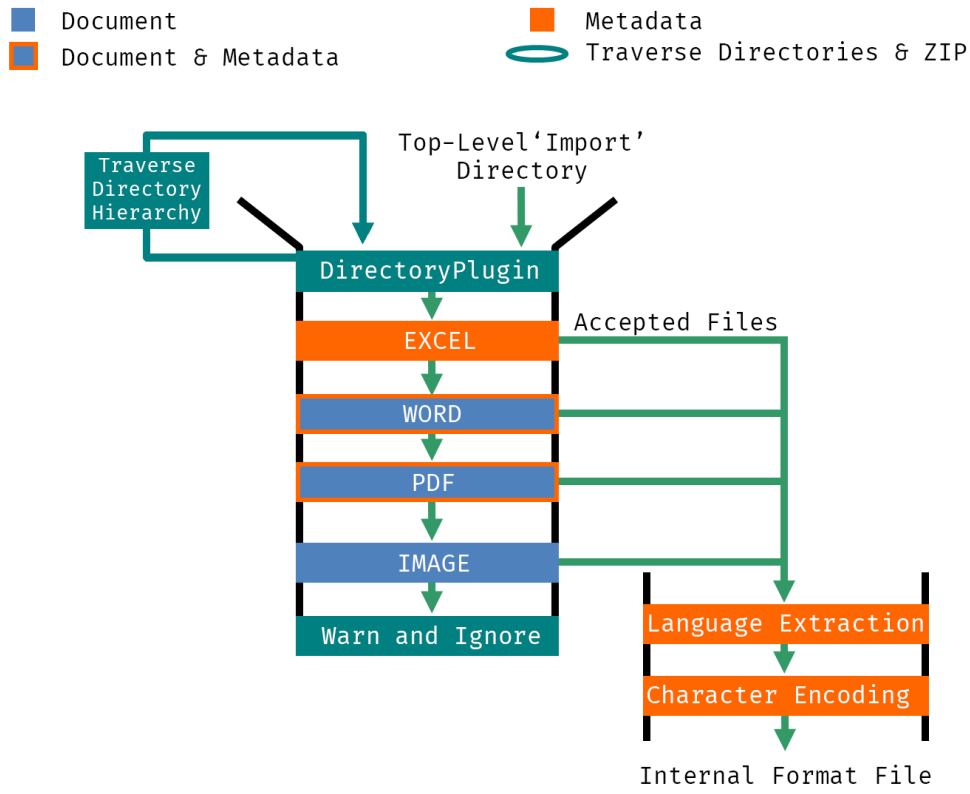

Figure 2.9: Greenstone 3 Plugin Architecture

metadata to files based on the content of a spreadsheet. If the spreadsheet contains a file name row with an entry matching the input file name, the *EXCEL* plugin can extract and attach information from the corresponding columns. This is useful in cases where the digital librarian wishes to supply metadata in the form of a spreadsheet, whether it be manually or automatically generated.

The *WORD* and PDF plugins manage the import of Microsoft Word and PDF documents respectively, collecting available metadata such as the author and title from the property headers of these file formats. The *IMAGE* plugin ingests image files and does not extract any metadata but can perform tasks such as converting the original file to another image format, along with thumbnail and screen-view-sized versions for users in the digital library.

In the event a file passes entirely through the pipeline with no plugin signalling that it has processed the file, then the *Warn and Ignore* block at the end of the pipeline ensures the digital librarian is aware that there were input

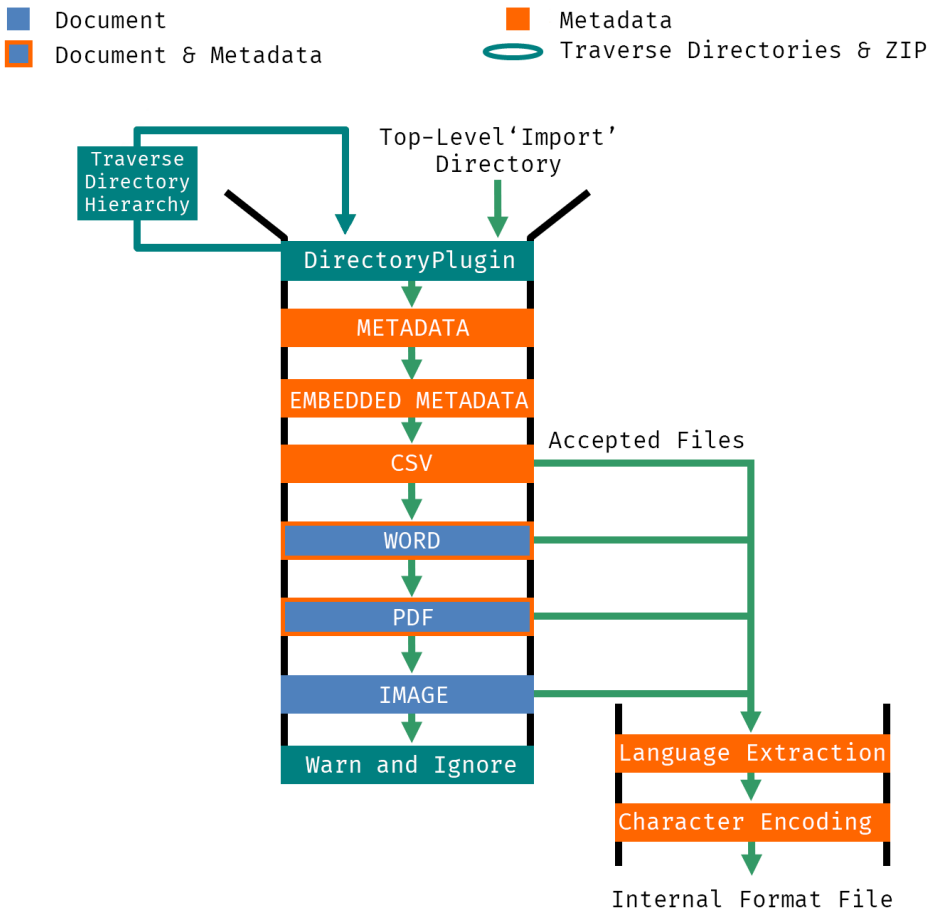

Figure 2.10: Greenstone 3 Plugin Architecture Extended (*Note:* If an import file is not matched with any of the plugins, a warning is given to the digital librarian and the next import item is ingested into the pipeline.)

files not accepted by any plugin.

This introductory plugin pipeline example can be expanded to allow for more sophisticated connections using specialised plugins where further metadata fields are accessed and extracted. Figure 2.10 illustrates some additions that expand the sources from which metadata is accessed.

The *METADATA* plugin in Figure 2.10 attaches manually assigned metadata stored in *metadata.xml* files to other files in the import directory which are subsequently processed as a document by another plugin. To complement this, the *EMBEDDED METADATA* plugin extracts embedded metadata, such as EXIF, from numerous file types including MPEG, JPEG, and HTML. The added *CSV* plugin attaches metadata based on values within a CSV file. It

works similarly to the *EXCEL* plugin, however, in this example it has been configured so in the special case where the header to the CSV file contains the *Filename* label—and similar to how the *METADATA* plugin works—this field is used to determine which other file in the import directory the specified metadata should attach to. Note that the *EMBEDDED METADATA* plugin itself never signals that it has accepted a file for processing. This is due to the plugin extracting any metadata present and then moving the document down the pipeline. The inclusion of a CSV file is an effective method of providing extra information to support the primary documents.

The import process also involves the assignment of associated files to the standardised documents that are being formed. In Greenstone, an associated file refers to a supplementary file that is linked or related to a primary digital document, often used to provide additional context or resources such as images, metadata, or related documents.

The result of Greenstone's import stage is to generate a top-level folder called *archives*. In this directory, there is one folder per document created. So as to not over-burden the file system with too many documents all contained in the same folder, *archives* is organised into a hierarchical series of subdirectories.

Build

Building a collection is comprised of indexing the documents in the new *archives* directory and generating the full-text indexes along with a database of metadata and classifier structures, resulting in readily accessible information for efficient document and metadata retrieval. Associated files, created indices, and the generated database are temporarily stored in a directory named *building*. The *building* directory is removed after the build process, while the others remain in the collection's directory. This allows the archival and "live" versions of the documents to be retained and accessible, ensuring that users can still access the previously built digital library collection while a new build process is running.

Activate

The activation process moves the content from the *building* directory to the *index* directory—removing any previous version if it exists—at which point the collection is now accessible online (Osborn et al., 2011).

2.3.2 Extensions

Greenstone's extension mechanism is an expanding set of improvements often developed as part of research projects. Such extensions expand the capabilities of Greenstone, usually through the use of external tools. Greenstone currently has the ability to enrich image documents by applying Google Vision API processes and to enrich audio documents by extracting diarisation data using Pyannote.

Google Cloud Vision API

Google's Cloud Vision API is a cloud-based service that provides image analysis capabilities through the use of machine learning and computer vision technologies. Key features include object classification, OCR, and face detection (Omena et al., 2021). Greenstone supplies an option to enable Google Cloud Vision integration, which extracts text from images ingested into a collection. Figure 2.11 shows an image document in a Greenstone collection—available publicly on the Ātea website[13]—that has had OCR performed at ingest using the Google Cloud Vision API. Extracted text can be searched for, enriching the discoverability of image-based documents.

Pyannote

Pyannote provides a set of trainable end-to-end neural building blocks that can be combined and jointly optimised to build speaker diarisation pipelines (Bredin et al., 2020). Speaker diarisation is the task of determining "who spoke when?" in an audio or video recording that contains an unknown amount

[13]https://atea.space

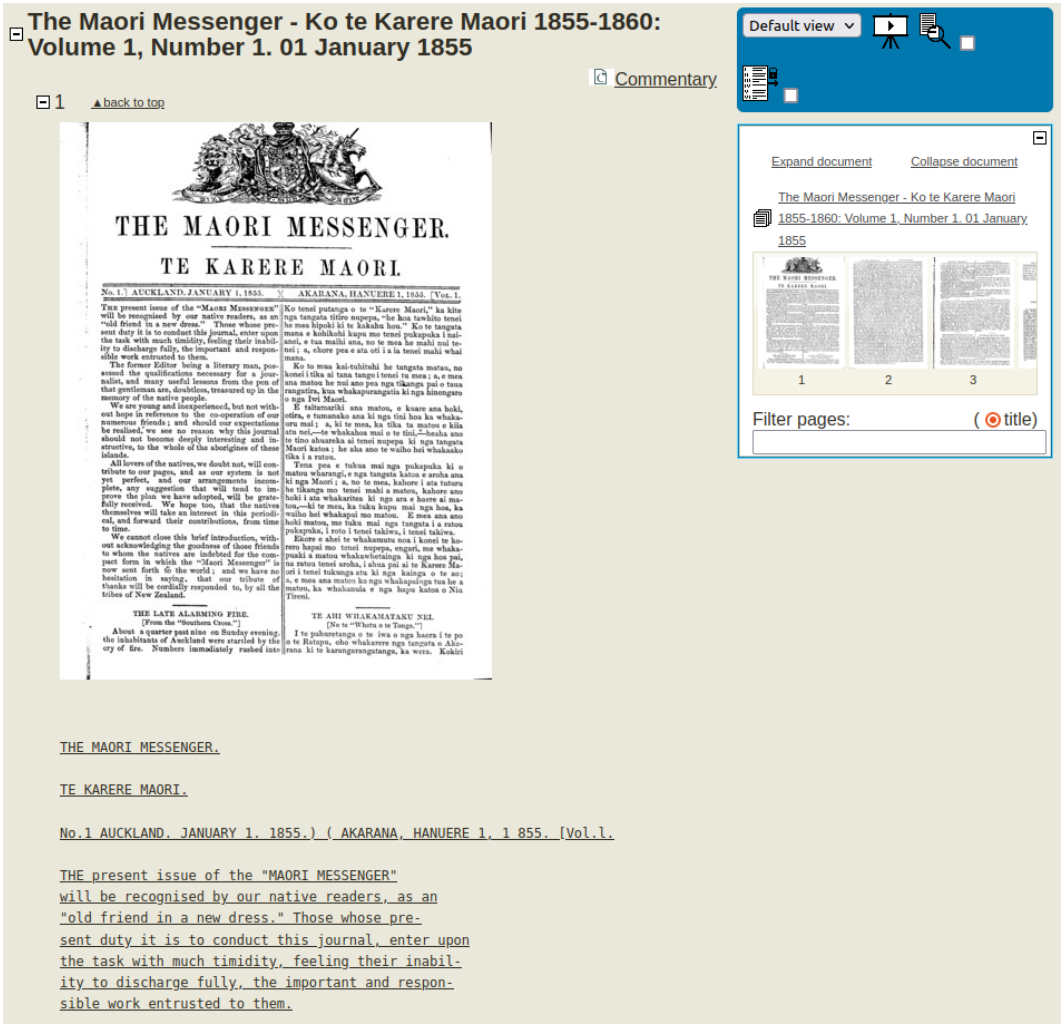

Figure 2.11: Greenstone 3 Google Cloud Vision Integration

of speech and also an unknown number of speakers (Anguera et al., 2012). Pyannote is an open-source toolkit written in Python. Greenstone offers the option to integrate Pyannote into a digital library collection containing audio, which visualises the diarisation data with an improved audio player.

Figure 2.12 shows the existing audio player with speaker regions generated by Pyannote in its state before any additions were made as part of this thesis. In preparation for this thesis, the author developed the audio player under Dr. Bainbridge's supervision as part of COMPX397 at The University of Waikato. This involved developing a Greenstone plugin to ingest audio files, utilising Pyannote to generate diarisation data, and then developing a web interface that displays the speaker regions over a waveform. Notably, it did not

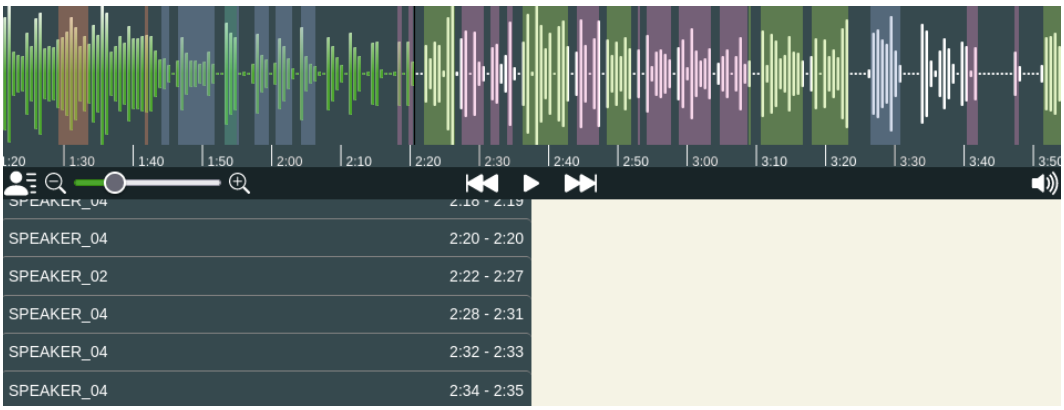

Figure 2.12: Existing Audio Player in Greenstone (*Note:* Audio player in a Greenstone collection with diarisation data on waveform and chapter list.)

include any means of user feedback or editing.

When a document is ingested by the Greenstone Digital Library, through the use of a custom plugin *StructuredAudioPlugin*, the diarisation data is automatically generated by Pyannote in the form of a CSV file when a collection is built, providing a seamless acquisition of diarisation information which can subsequently be attached to the corresponding document. Speaker identity is represented with integer values, which aids in recognising same-speaker segments but does not give the ability to associate an actual name. The audio player is built using Wavesurfer.js, an open-source JavaScript library that provides customisable audio waveform visualisation, additionally supplying audio navigation functions such as play, pause, and volume interactions[14]. Wavesurfer.js also provides a plugin that allows regions to be created on the waveform, which is how the diarisation data extracted using *StructuredAudioPlugin* is represented. Segments can be navigated by clicking either on the waveform or in a list of segments. The user interface will be further described in Chapter 5.

When a Greenstone collection is built and a flag is specified indicating audio in the collection should be processed through Pyannote, *StructuredAudioPlugin* generates a CSV file containing columns for speaker names, start times, and

[14]https://wavesurfer-js.org/

stop times. The CSV file is linked to the corresponding audio document as an associated file. The start and stop values are represented in seconds rounded to one decimal place.

```
SPEAKER_00, 0.5, 4.0,
SPEAKER_01, 7.0, 9.5,
SPEAKER_00, 12.2, 15.0,
```

Pyannote is overlap-aware, meaning it can detect and output overlapping speaker segments (Bredin and Laurent, 2021). Overlapping speaker segments are represented in the Greenstone audio player waveform by overlapping coloured squares.

2.4 Discussion

Motivated by the work of this thesis, the Greenstone digital library development team expanded Greenstone's capabilities by allowing for multiple versions of a single document to be stored. This is an integral element required in the context of the aforementioned topics, notably user corrections and content analysis. Greenstone's expandability through plugins, the ability to run content analysis tools at ingest, the relation to the University of Waikato, and its recent developments to support document versioning make it a prime framework upon which to base the technical work for this thesis.

In the next chapter, a literature review is conducted to establish existing published works related to the realm of digital libraries, content analysis, user corrections, and content analysis output combination. This sets the stage for the remainder of the thesis, introducing concepts and algorithms that will be expanded in later chapters.

Chapter 3

Literature Review

3.1 Improving Content Analysis

This chapter explores strategies for enhancing content analysis techniques using user input. While user effort is generally expensive, it is often of high-quality, making it a good candidate for improving automated content analysis system outputs. Human data can be integrated through machine learning techniques or by more direct approaches. Such strategies will be outlined in general environments and digital library environments.

3.1.1 Human-In-The-Loop

Human-in-the-loop (HITL) refers to a concept in machine learning where human expertise and intervention are integrated into automated systems to enhance their performance or quality control. HITL implementations have been developed for OCR (Benedikt et al., 2020), diarisation (Prokopalo et al., 2021), image classification (Budd et al., 2021), and many more content analysis types. These implementations use manual interventions to iteratively train and calibrate models, improving the result.

To make human-machine interactions more user-friendly, specialised interfaces are often designed to make gathering manual data easier for a wider range of users. Figure 3.1 shows a layout proposed by Benedikt et al. which

Figure 3.1: OCR HITL User Interface Design Example [Reproduced from (Benedikt et al., 2020)].

allows for the correction of fields to aid the OCR process on images of receipts. This particular example is from a system that passes samples onto human correctors when there is a low confidence score of the initial classification stage. This is a common component in HITL systems as it ensures human efforts are channelled towards samples that require attention in order to be improved to a reasonable confidence threshold.

The goal of HITL is to improve the output of an automated process through human interjections. Such interjections tend to result in a positive impact on the final result. However, the exact edits may not appear in the output, even if the edits made an overall positive impact on the output.

3.1.2 Crowdsourcing

Crowdsourcing is a type of participative online activity in which "a host entity proposes to a group of individuals of varying knowledge, heterogeneity, and number, via a flexible open call, the voluntary undertaking of a task" (Estellés-Arolas and González-Ladrón-de-Guevara, 2012). The user usually receives satisfaction of some type, while the host obtains some form of work from the user. This is an intentionally loose definition as crowdsourcing can be

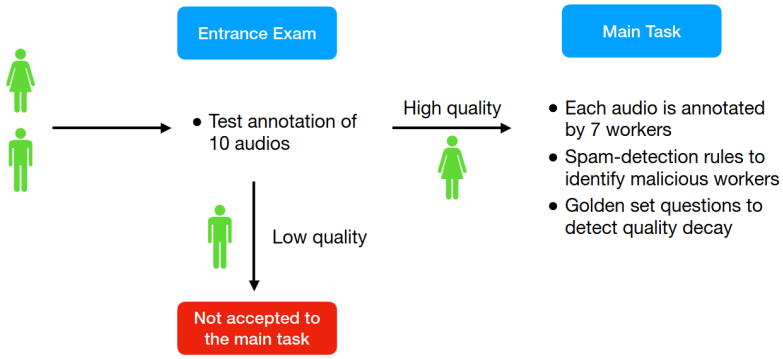

Figure 3.2: Entrance Exam for Crowdsourcing Project [Reproduced from (Pavlichenko et al., 2021)]

implemented in many environments.

Examples of crowdsourcing implementations include reCAPTCHA, Patreon, Kickstarter, and The Snapshot Serengeti Project. The Snapshot Serengeti Project is a wildlife monitoring initiative that utilises camera traps placed throughout the Serengeti National Park in Tanzania. To classify which animal was present in each of the images taken, volunteers participated through a web interface to identify and label each sample. Collectively, over 28,000 registered users contributed 10.8 million classifications (Swanson et al., 2015).

Addressing large-scale problems such as animal classification with crowdsourcing is an efficient method of obtaining large amounts of high-quality data. Undoubtedly, due to its involvement with the public, not all entries will be completely accurate. However, authentication, validation, and plurality voting help ensure the correct resolution is reached.

Ensuring user tasks are designed effectively will promote high-quality inputs. This includes flagging users who are, regardless of intent, providing low-quality data. Figure 3.2 by Pavlichenko et al. illustrates a simple preliminary stage where users who fail to annotate a set of questions with known answers are not accepted to the actual annotation task. Such questions are often referred to as *golden set* questions or verification tasks, where the goal is to ensure the user is providing correct answers before the real tasks are given.

Incentive is another crucial element of a crowdsourcing system. If there

is no incentive, users will not provide their inputs and the desired data will not be obtained. In The Serengeti Project, participants contributed to science and conservation, learned about Tanzanian wildlife, and potentially fostered a sense of community among other volunteers sharing a common interest. In other cases, leaderboards are implemented to incentivise a drive for a high rank and monetary incentives may also be used in cases where specialist input is essential.

3.1.3 User Corrections in Digital Libraries

Digital collections are increasingly used for purposes often motivated by qualitative and quantitative research (Chiron et al., 2017). Giving users the ability to correct metadata within digital collections not only improves search precision, but also engages more people with the content and enriches the quality of documents.

CiteSeerX, as discussed in Chapter 2, is a public digital library and search engine for academic papers, hosting over ten million documents. For a given document, CiteSeerX offers the ability to correct the title, abstract, authors, and other attached metadata for registered users. A *details* field is required so that edit justifications can be given. Error corrections are manually reviewed by staff before they are published. The moderation stage ensures edits are of a high-quality and mitigates the effects of spam. A study was published that investigates the impact of user corrections on the number of downloads a document has in CiteSeerX. It is proven that on average, user corrections led to a 25% increase in the number of times a document is downloaded (Wu et al., 2014).

Trove

As briefly outlined in Chapter 1, from the National Library of Australia, Trove is a search engine with a primary focus on aggregating Australian content. The Trove database includes newspapers, images, maps, video, and audio (Holley,

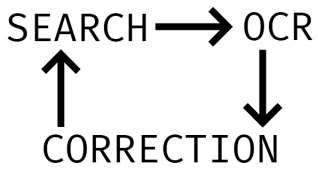

Figure 3.3: User Corrections Improve Search Quality

2010). Trove gives users the ability to correct the text transcript of library items. A Trove account is not needed to make edits and edits appear in search results within five minutes after submission. Each user edit overwrites any previous edit on the same text, with moderation being managed by the Trove community. The edit history is saved so that inaccurate edits can be reverted. Satisfaction is provided to users in the form of points towards a correction leaderboard present on the Trove website.

Figure 3.3 is an excerpt from a Trove crowdsource performance review illustrating the role user corrections have within a digital library (Hagon, 2013). With more accurately represented data, searching in a collection will return more relevant results, allowing users to more easily access the data they are looking for.

Veridian

Veridian is a service offering digital collection and digitisation services to aid in the preservation of historic content. A User Text Correction (UTC) module is offered which allows users to correct OCR errors as they come across them in the text. There is also an audio transcription module present in Veridian's "Kani'āina – Voices of the Land" Digital Library. Both OCR and transcription correction improve the searchability of digital archives and help to better represent historical and cultural data.

To correct OCR text in Veridian, a user must register an account. Any changes saved by the user are visible immediately. Edits are made in blocks of text, where a checkbox containing the text: "This block is completely correct"

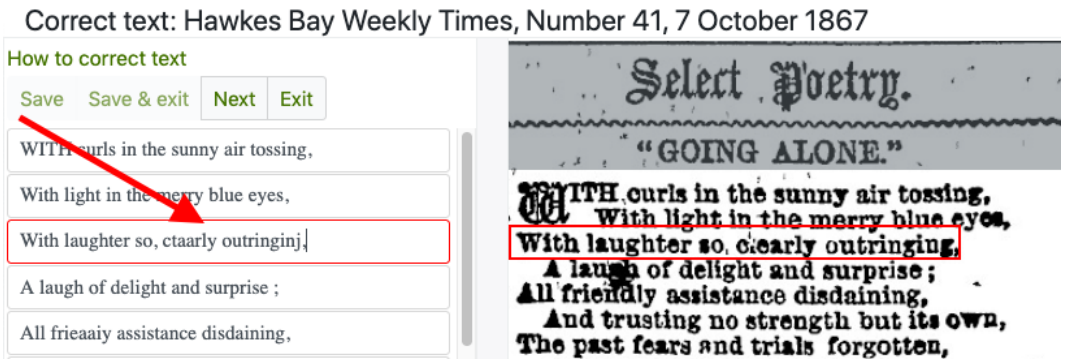

Figure 3.4: UTC in Veridian (*Note:* User text correction in a Veridian digital collection. Users donate their time, correcting OCR errors in textual documents. [Reproduced from ("Veridian User Text Correction", 2023)])

is offered, indicating whether or not the user has checked over all of the text in the block. This feature helps build UTC statistics for each collection and for other users to see where someone has already finished a section ("Veridian User Text Correction", 2023). Figure 3.4 shows the UTC module in use on an excerpt from the Hawkes Bay Weekly Times. Editable text blocks are offered one line at a time to simplify the process and avoid newline discrepancies.

Digitalkoot

Developed to index the National Library of Finland's archives for improved searchability, Digitalkoot is a gamification of the OCR correction process. In a sample article containing 1,467 words, Digitalkoot produced only 14 mistakes, a <1% error rate (Chrons and Sundell, 2011). Known verification tasks, also present in reCAPTCHA, were employed to monitor user performances and keep up the gameplay. Overall, the project achieved over 99% accuracy through almost 5,000 users pitching over 2,740 hours. One of the gameplay modes is shown in Figure 3.5.

Limitations

A limitation present in digital libraries incorporating user input is quality control. Quality control is an important component of a crowdsourcing system

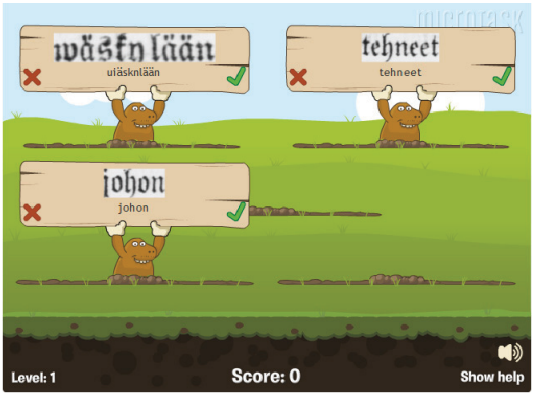

Figure 3.5: OCR Verification in Digitalkoot (*Note:* Gamification is an effective incentive for users to complete tasks. [Reproduced from (Chrons and Sundell, 2011)])

to ensure bad inputs are filtered out. The first layer of defence commonly implemented is an authentication layer. This ensures users have created an account and optionally have been through a screening process to limit the number of spam accounts.

Verification tasks such as those present in reCAPTCHA ensure the quality of a user's answer is legitimate and holds good intent. A verification task is a question given to a user with a known answer. If the user answers the question correctly, it can be assumed that other answers given by that user have good intent. Tasks involving the public can often attract users with ill intent. Ensuring multiple policies are in place to mitigate bad data is crucial to produce the best possible output.

Another limitation in crowdsourcing systems is if users do not relate to or hold an interest in the focal data, motivation and engagement can be challenging issues. Without incentives or rewards, contributors may lose interest in providing their efforts. In the realm of digital libraries, it could be assumed that the user has interacted with a particular document because they already hold an interest in it, however, more motivation will likely result in more corrections.

3.1.4 System Re-Run

In the lifetime of a document within a digital collection, a digital librarian may decide to re-run the automated process to extract better data. Adjustments made to justify a re-run include the content analysis system, for example Pyannote, the Python module, incurring a software update, or the adjustment of diarisation parameters. This process would likely be motivated by the potential improvement in the quality of the existing diarisation data. Alternatively, an entirely different content analysis system may be introduced.

In the case of diarisation, another situation where the process may need to be re-run is if the respective audio undergoes improvements such as noise reduction. Studies have shown that noise reduction and echo cancellation improves diarisation performance (Hain et al., 2006; Wooters and Huijbregts, 2007). Similar dualities could be realised in environments employing other content analysis systems such as OCR or transcription. OCR and other image-based content analysis approaches benefit from parameter adjustments and data preprocessing. Preprocessing is considered an integral step during the development of an OCR system, as it can vastly improve noisy or otherwise imperfect inputs (Dey et al., 2022).

3.2 Combination of Content Analysis Outputs

Merging multiple outputs from content analysis systems helps to improve accuracy rates and reduce errors. The process is exercised in OCR, diarisation, speech recognition, as well as speech and natural language processing tasks. Such concepts have been used for many years in other areas such as weather forecasting, where multiple independent predictions are combined to produce a more accurate forecast. In many fields, "combining can reduce errors arising from faulty assumptions, bias, or mistakes in data" (Armstrong, 2001).

The following section outlines combination systems that aim to improve accuracy rates over several content analysis fields.

3.2.1 ROVER

The ROVER system aims to reduce error rates for Automatic Speech Recognition (ASR) engines by aligning multiple ASR outputs and generating a consensus using minimal-cost word transition networks and a voting process (Fiscus, 1997). Multiple sequence alignment (MSA) is used to align multiple ASR outputs to identify their corresponding positions and similarities. It is commonly utilised in bioinformatics to align genes or proteins to find evolutionary relationships (Carrillo and Lipman, 1988).

The voting method uses frequency and confidence values to select the best-scoring word sequence. As stated in the paper, "it is not always the case that ROVER is able to develop the best scoring hypothesis for a segment. In some cases, only one of the systems has the correct hypothesis, and its hypothesis is out-voted by more errorful systems" (Fiscus, 1997). In the context of a digital library, this concept is mitigated by the combination being a manually instated action. If one of the systems has the correct hypothesis, there would be no incentive to achieve a better result through combination.

3.2.2 DOVER

Developed by Microsoft's Speech and Dialog Research Group, the DOVER system combines multiple diarisation outputs to improve accuracy (Stolcke and Yoshioka, 2019). DOVER operates in two segments: label mapping and consensus voting.

Because the inputs into DOVER are likely to contain different speaker names, the label mapping method seeks to map the anonymous speaker labels into a common label space. The method works by finding a mapping between two diarisation outputs that minimises the total time duration of speaker mismatches. This pairwise operation is repeated across all inputs incrementally until all speaker regions are represented by a common label space.

The voting segment uses majority voting to derive a consensus output. This tends to result in unifying similar regions from parallel diarisation sys-

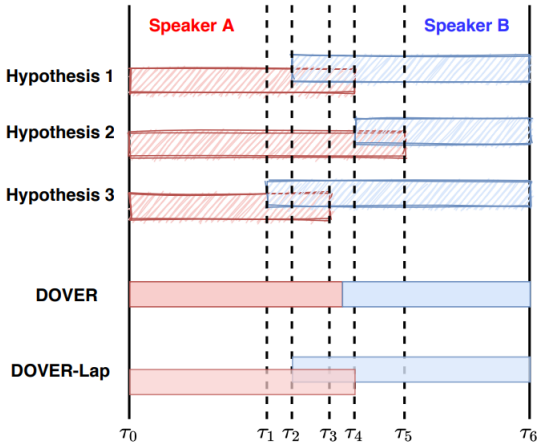

Figure 3.6: DOVER-Lap vs. DOVER Comparison (*Note:* Overlapping speaker detection can be useful to represent audial interjections within a conversation. [Reproduced from (Raj et al., 2021)])

tems. DOVER is found to consistently beat the average accuracies of the input channels.

A duality can be observed between ROVER and DOVER through the alignment requirements. In ROVER, the input (word) labels already live in a common namespace (the vocabulary) and need to be aligned in time. In DOVER, the input (speaker) labels live on a common time axis and need to be aligned in a common namespace (mapped) (Stolcke and Yoshioka, 2019).

DOVER-Lap

Inspired by DOVER, DOVER-Lap is designed to support overlapping regions in the combination of diarisation outputs. The incremental pairwise mapping is replaced with a global mapping strategy based on a cost tensor and overlap awareness in label voting is introduced. This strategy allows multiple overlapping speakers to be present in the combined diarisation output (Raj et al., 2021).

Figure 3.6 illustrates the key difference in the outputs generated from DOVER and DOVER-Lap. The inability to produce overlap-aware diarisation outputs is DOVER's most significant drawback. DOVER-Lap mitigates

this issue, however, brings forward another problem: it is built to combine more than two diarisation system outputs. Due to their voting strategy, it is stated that operating with two inputs will almost certainly lead to performance degradation (Raj, 2023).

3.2.3 OCR Combination

Modern OCR systems can achieve good results when given high-quality input documents. In many contexts, including digital libraries, the quality of the documents cannot be guaranteed, nor easily improved. Common strategies for improving the output of OCR systems include preprocessing steps such as input image adjustments and introducing dictionaries to the OCR process. These are often coupled in the OCR system. For example, Tesseract, a popular open-source OCR engine, applies various image processing operations internally to improve the quality of output (Tesseract-OCR, 2023).

Combining multiple OCR outputs is another way to increase accuracy rates. There are two approaches for combining OCR outputs: combining the outputs of multiple OCR systems given the same input and combining the outputs of one OCR system given multiple versions of the same input. These each have their use cases and bring forward the issue of text alignment.

Ocromore

Ocromore is a toolset used to increase character recognition accuracy by combining multiple OCR results (Kamlah and Stegmüller, 2018). It is built for combining multiple OCR outputs which have been given the same input. While offering an entire OCR pipeline with a preprocessing stage, Ocromore more notably introduces an alignment strategy to position OCR outputs so that differences can be observed at the character level.

The alignment strategy starts by using bounding boxes on each line and word to identify corresponding areas of the OCR outputs. For character alignment, a Multiple Sequence Alignment (MSA) algorithm was employed, much

like the aforementioned ROVER. Such algorithms are typically used to arrange biological sequences found in DNA, RNA, and proteins (Edgar and Batzoglou, 2006). Using the edit distance between strings to gauge the similarity and "wildcards" to fill in missing characters, the characters between multiple OCR outputs can be aligned. This ensures the voting process gathers an accurate representation of the OCR data to derive a consensus.

Ocromore uses a voting process that utilises confidence scores to derive a consensus output. Each input is weighted based on its overall confidence score. The confidence score of each character is scaled using the input weights and then the character with the highest confidence is chosen as the consensus.

Using results from the Abbyy, Ocropus, and Tesseract OCR engines, the results show an accuracy increase of 0.49% when compared to the best single result, as well as an error reduction of 33% (Kamlah and Stegmüller, 2018).

Concept by Wemhoener et al.

Wemhoener et al. introduce an automated scheme for aligning and combining OCR output from three scans of a book to generate of composite version with fewer OCR errors (Wemhoener et al., 2013). Similar to the label mapping process in DOVER, this approach uses a "pivot" input to align with each of the other inputs. The pairwise alignments are made using a recursive text alignment scheme (RETAS) introduced in another study (Yalniz and Manmatha, 2011). A multiple sequence alignment approach (also utilised in Ocromore) then aligns the inputs to the character level, ready for consensus voting. The voting process is driven by a majority vote—whichever character occurs most is chosen as the consensus.

The method outlined in Ocromore is an example of multiple OCR systems processing the same input. The output from each of the systems is then combined. Conversely, the approach described in Wemhoener's paper puts multiple versions of the same input through the same OCR system. The latter requires an alignment scheme that can handle extra portions of text at the

beginning or end of the inputs. It is stated that although one of the input texts contained a large amount of extraneous text and another was missing a significant portion of the text, the composite text is more accurate than any of the editions and it includes the complete copy of the original work (Wemhoener et al., 2013).

3.2.4 Further Applications

Output combination with the goal of improving error rates is theoretically beneficial in any content analysis approach as it aims to combine the best of multiple outputs. Output combination frameworks have been proposed for machine translation (Chen and Chen, 2009), handwritten text transcription (Granell and Martínez-Hinarejos, 2015), automatic speech recognition (Fiscus, 1997; Sinha and Siegert, 2021), and diarisation (Stolcke and Yoshioka, 2019). All report improvements in error rates over the best-scoring input and most incur an alignment phase followed by majority voting.

3.3 Discussion

Diarisation is a growing form of content analysis. Its increasing appearance in citations can be seen in Figure 3.7. Output combination through the use of scripts such as ROVER and DOVER is a reliable method to improve error rates, as proven in the respective papers (Fiscus, 1997; Stolcke and Yoshioka, 2019).

Re-runs of Pyannote and the combination of content analysis outputs are both instances where user edits may be lost. With the ability to edit documents in a digital library, ensuring these edits are retained is important to both the editor and quality of the document.

A drawback to the ROVER derivatives and the aforementioned OCR combination systems is that inputs containing user corrections lose those corrections during the merge. DOVER, DOVER-Lap, and Ocromore offer the ability

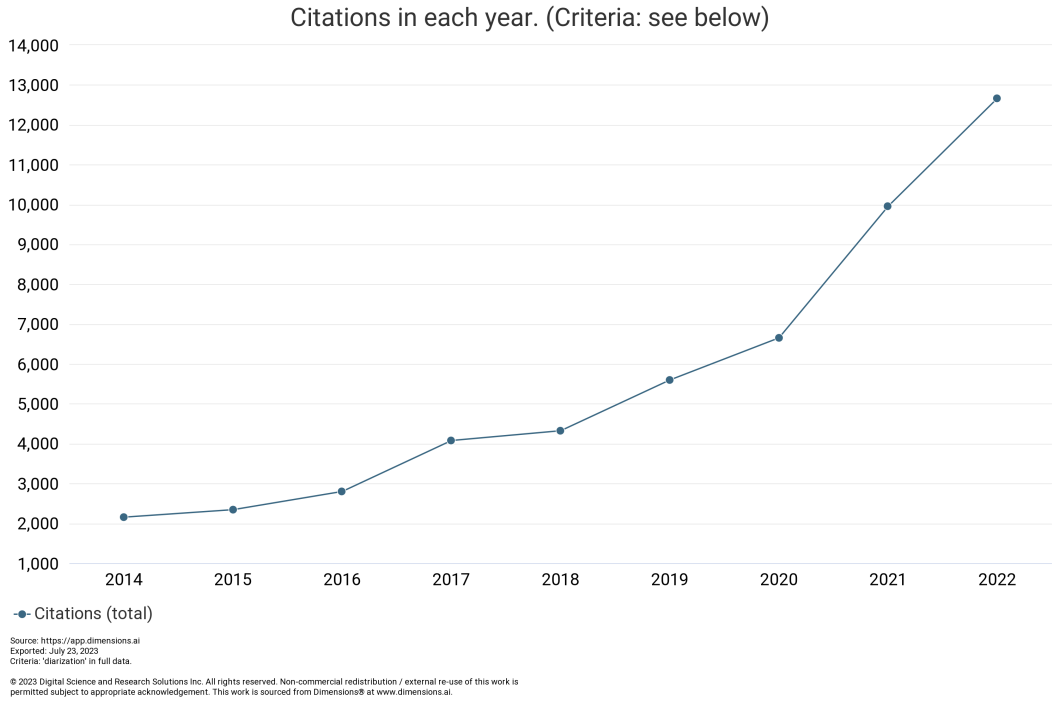

Figure 3.7: Citations of Publications Including the Term "diarization" [Reproduced from (Hook et al., 2018)]

for weights to be applied to each input. This is useful in scenarios where particular systems are known to be of higher quality, however, does not properly control cases where the user wishes for the user edits to be retained and the remaining content to be merged.

The review of existing digital libraries and content analysis techniques identified the limitations of user edit retention in existing output combination methods. The concept of loss of user edits was made apparent, setting the stage for the remainder of this thesis.

Chapter 4

DOVER-Lock

4.1 Introduction

This chapter details the development of a new algorithm: DOVER-Lock. DOVER-Lock is designed to address the limitation of its namesake, DOVER, where all input speaker segments are treated equally in determining which segments are output. In the situation where the result of manual corrections is included as input as one of the speaker segmented tracks, in DOVER, this can have the detrimental effect of eliminating segments that have been manually corrected. The motivation for the development of DOVER-Lock is to reduce error rates while retaining high-quality manual corrections.

The concept of locking edited sections of content is introduced in a manner which means the general approach is transferrable to other disciplines such as OCR and further content-analysis systems that benefit from output combination. DOVER is explored in depth and the core scripts of the reference implementation made available by the author of this work are examined. Then, DOVER-Lock is introduced and described followed by an analysis and discussion of the results.

Table 4.1 shows which of the previously reviewed content analysis output combination systems support individually weighted inputs and the lack of support for particular sections to be locked or weighted and assesses them with

regard to their capabilities to protect specific segments of the input provided. Neither ROVER nor the method by Wemhoener et al. has any capability for weighted inputs, however DOVER, DOVER-Lap, and Ocromore do. As mentioned previously, weighted inputs are a useful addition to an output combination tool to give precedence to a higher quality input, however, does not address weighting on a per-section basis. Per-section weighting is applicable to multiple forms of content analysis output combination and would allow the retention of high-quality user corrections when outputs are combined.

	Weighted Inputs	**Weighted or Locked Sections**
ROVER	No	No
DOVER	Yes	No
DOVER-Lap	Yes	No
Ocromore	Yes	No
Method by Wemhoener et al.	No	No

Table 4.1: Output Combination Weighted Input Support

4.1.1 DOVER

The official implementation of DOVER is hosted on GitHub[1] and is comprised primarily of Bash scripts. An externally sourced Perl script is also used for diarisation system evaluation. The DOVER method was developed by the Speech and Dialog Research Group of Microsoft. While DOVER is not a direct variant of ROVER, a duality can be observed between the two algorithms (Stolcke and Yoshioka, 2019). To recognise the work completed in the ROVER (Recogniser Output Voting for Error Reduction) algorithm, DOVER (Diarisation Output Voting for Error Reduction) is named after it.

[1]https://github.com/stolcke/dover

DOVER has been proven to reduce error rates. The generated consensus output "consistently beats the averages of the input channels, and can be very close or improving on the oracle error rate obtained by picking the single best channel for a given meeting" (Stolcke and Yoshioka, 2019). The algorithm first maps the input's speaker labels into a common label space. This aligns the speaker names and results in files denoting the speaker name matches. Following the label mapping, a consensus output is found by performing a simple voting process for each region in the audio.

Inputs and Outputs

Rich Transcription Time Marked (RTTM) files are used as both the input and output file types in DOVER. RTTM files are space-delimited text files used to contain a collection of time stamps and labels occurring throughout audio, with a speaker segment per line. A RTTM entry includes the following ten items: Type, File ID, Channel ID, Turn Onset, Turn Duration, Orthography Field, Speaker Type, Speaker Name, Confidence Score, and Signal Lookahead Time (Ryant et al., 2018). The following is an example of two speaker segments:

```
SPEAKER Meeting_1 1 6.5 7.2 <NA> <NA> SPEAKER_01 <NA>
SPEAKER Meeting_1 1 21.0 4.1 <NA> <NA> SPEAKER_02 <NA>
```

The tenth column, representing signal lookahead time, is occasionally omitted in literature as well as being completely excluded throughout DOVER and its corresponding literature. Because of this, the RTTM files used throughout this project will be represented by nine columns.

The DOVER script, along with the consensus output RTTM, generates several log and helper files which are stored in a directory (by default named *DOVER*) in the current working directory when the program is run. Examples of files generated include map files which record the speaker label mappings and list files which store the file names of all inputs. Files are written to and read from this directory throughout the DOVER runtime process.

md-eval.pl

The standard code used to evaluate speaker diarisation systems is *md-eval.pl*, provided by NIST as part of its scoring toolkit[2] (McKnight et al., 2021). This Perl script is used to calculate the diarisation error rate between inputs for label mapping as well as logging region overlap durations. The overlap durations are also used in the label mapping stage to minimise the mismatch between two diarisation label sequences.

The accuracy of a diarisation tool is measured using the diarisation error rate (DER):

$$DER = \frac{ERR + FA + MISS}{Total\ Duration}$$

where *ERR* is the total speaker time represented by the wrong speaker label, *FA* (False Alarm) is the total time of the falsely detected speech, and *MISS* is the total time of missed speech detection. DOVER executes *md-eval.pl* on two RTTM files at a time. One is the comparand reference RTTM and the other is the file to compare to the reference. This comparison is applied to all combinations of inputs such that an example with three RTTM inputs $\{A, B, C\}$ would call *md-eval.pl* with $\{B, A\}$, $\{C, A\}$, and $\{C, B\}$ as input parameters.

Label Mapping

The DOVER algorithm starts by mapping the anonymous speaker labels into a common namespace. This is required as different OCR systems will likely follow dissimilar speaker name conventions. For example, Pyannote, the diarisation system used in Greenstone, writes speakers as "SPEAKER_XX", where "XX" is a number starting at "00". Pyannote generates the speaker names incrementally, adding one to each new speaker identified. NVIDIA's NeMo[3] is a toolkit containing a speaker diarisation tool which outputs speakers as

[2]https://github.com/usnistgov/SCTK

[3]https://github.com/NVIDIA/NeMo

"speaker_X" (Kuchaiev et al., 2019). Different diarisation tools will use different naming conventions, meaning speaker names need to be mapped to a common namespace before proceeding to speaker voting.

Provided two RTTM inputs with speakers labelled $A_1, A_2, ..., A_m$ and $B_1, B_2, ..., B_n$ respectively, label mapping seeks to minimise the overall duration of inconsistencies between speakers and incorrect matches between speech and nonspeech elements from $\{A_i\}$ to $\{B_j\}$. For instances where more than two inputs are provided, the mapping is generated incrementally by mapping the third output to the first two after the second output is mapped to the first. This incremental pair-wise mapping is repeated until all RTTM inputs have been incorporated.

In the event all inputs are weighted equally, all inputs are mapped to the namespace of the first input. For an example set of files *ex-in1.rttm*, *ex-in2.rttm*, and *ex-in3.rttm*, this is the configuration that has been used to combine these speaker segment tracks shown in Figure 4.1. This figure and subsequent ones are produced using a NodeJS viewer[4] that was implemented as part of this work. The chosen namespace can be controlled by setting the weight of the input set with the desired speaker names through command-line options given when DOVER is run. This, however, does not give per-region control over which speaker names to keep.

Speaker Voting

To generate a consensus RTTM output, the speaker voting algorithm utilises a file named *collated* which contains all speaker segments from all RTTM inputs with the newly mapped speaker labels. Consensus regions are generated by "performing majority voting at each time instant (including on whether there is speech or not)" (Stolcke and Yoshioka, 2019). Figure 4.1 demonstrates how two regions with the same start or end time take precedence over the adjacent region. This can be seen in the rightmost regions where speakers A and I in

[4]https://github.com/finn-i/rttm-viewer

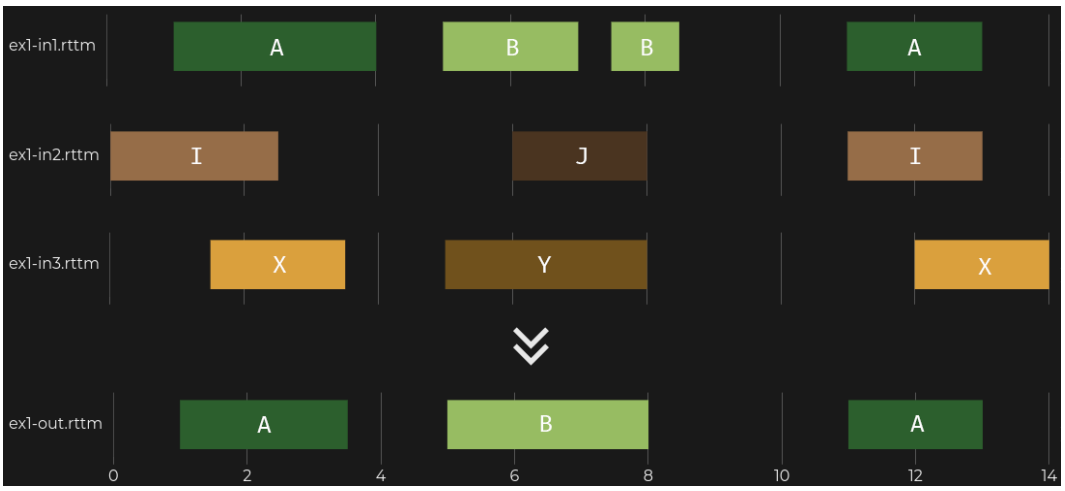

Figure 4.1: DOVER Label Mapping and Speaker Voting Example (*Note:* The speaker's name is shown by the colour of the region box and the letter in the region. In this example, the regions are merged fairly while retaining the speaker names present in the first input.)

ex1-in1.rttm and *ex-in2.rttm* respectively share the same start and end time. This takes priority, due to majority voting, over speaker *X* in *ex1-in3.rttm*.

4.2 DOVER-Lock: Methodology

The title *DOVER-Lock* was chosen to represent the following methodology as it extends the aforementioned DOVER script to support the retention of locked regions in the consensus output. In this section, the DOVER script as found on GitHub[5] is extended to provide the relevant additions.

Firstly, the installation requirements are outlined, followed by a review and justification of each implemented concept making up the DOVER-Lock script. Additional helper scripts are then described. The helper scripts aid in cleaning the output files so that they are reusable and contain extra functional information.

[5]https://github.com/stolcke/dover

4.2.1 Prerequisites

Development of DOVER-Lock was undertaken on a GNU/Linux Operating System and also tested on Cygwin. It has the same command-line utility dependencies as DOVER, with *AWK* and *SED* being key requirements, along with an installation of Perl to run the NIST evaluation script *md-eval.pl*.[6]

With the DOVER-Lock scripts on your PATH, it can be executed by issuing the following command:

```
dover-lock input1.rttm input2.rttm ... > output.rttm
```

where any number of inputs (more than one) may be provided.

4.2.2 Lock Types

Two lock types are proposed and implemented to separately represent the preservation of properties related to diarisation regions. The first is a duration lock which indicates the start and stop time of a region have been edited or confirmed and should be retained. The second lock is a speaker lock, indicating the speaker name of a region has been edited or confirmed and should be retained. The two different locks are necessary as there are cases in diarisation editing where the region bounds are identifiable but the speaker name is unknown. Conversely, the bounds of a region may be uncertain but the speaker name is apparent.

In situations where both the region bounds and speaker name are edited or confirmed, both locks can be applied to a region to indicate both properties should be retained. The next section describes how the RTTM file headers are adjusted to accommodate for per-region lock status.

[6]The code for the developed algorithm is available at https://github.com/finn-i/dover-lock

4.2.3 Region Locking

The majority of the region-locking logic is defined in a *AWK* file named *detect-lock-overlaps*. The script *detect-lock-overlaps* is called with every unique pair of label-mapped RTTM inputs. For example, if {*EX1*, *EX2*, and *EX3*} are label-mapped RTTM files, *detect-lock-overlaps* is executed with {*EX1*, *EX2*}, {*EX1*, *EX3*}, and {*EX2*, *EX3*} as parameters.

The script checks if a locked region overlaps with a region from the opposite input, saving these overlaps with particular flags in an output file named *overlaps*. The output file is checked directly before the speaker voting algorithm so that the *collated* file can be altered based on the contents of *overlaps*.

To represent region locks, the 9-entry RTTM file is expanded to include three more fields in the tenth, eleventh, and twelfth columns. These fields are binary entries representing duration lock, speaker lock, and a lock that is true (1) if either of the previous locks are true:

```
SPEAKER Meeting_1 1 6.0 7.0 <NA> <NA> SPEAKER_01 <NA> 1 0 1
SPEAKER Meeting_1 1 9.0 3.0 <NA> <NA> SPEAKER_02 <NA> 0 0 0
SPEAKER Meeting_1 1 15.0 4.0 <NA> <NA> SPEAKER_01 <NA> 0 1 1
```

The third binary value is derived using a bitwise OR operation and is used in the DOVER-Lock script to give precedence to regions containing this particular lock when deriving the consensus output.

Duration Lock

If a region overlaps with a duration-locked region, it is printed in the *overlaps* file and flagged with a keyword for removal. As observed in the leftmost column of Figure 4.2, the duration-locked (identified with a red border) region by speaker *A* overlaps regions by speaker *I* and speaker *X*. These two overlapping regions are marked in *overlaps* for subsequent removal. This ensures the retention of the duration-locked region.

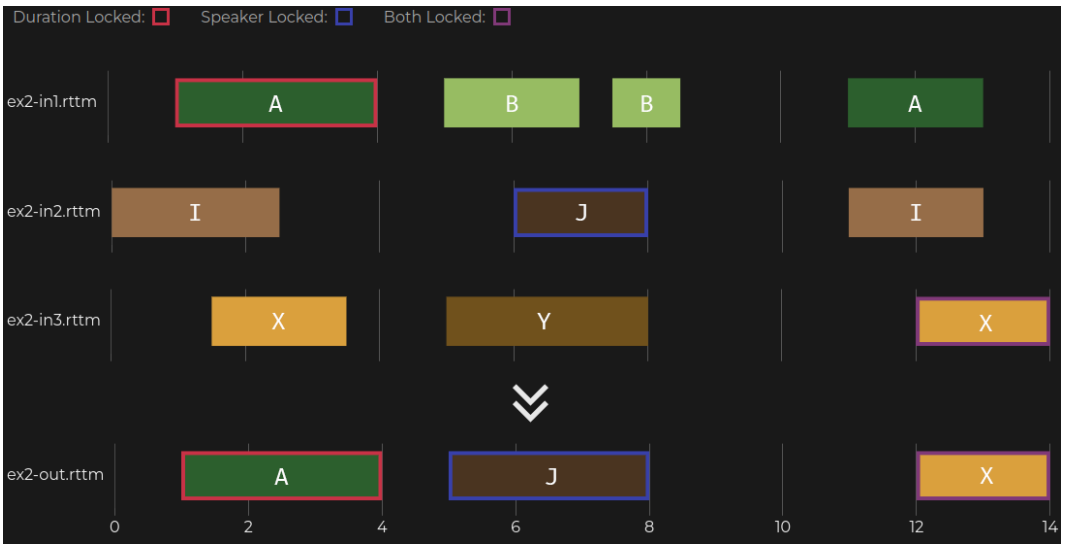

Figure 4.2: Region Locking Example (*Note:* A red border denotes a duration lock, blue denotes a speaker lock, and purple denotes both.)

The speaker name (in this case speaker A) is retained as usual. The pseudo code of the duration lock check in its most simple form is outlined in Algorithm 1.

Speaker Lock

If a region overlaps with a speaker-locked region, it is printed in *overlaps* and flagged with a keyword for a rename along with the name of the speaker-locked region. This allows for regions overlapping with a speaker-locked region to be renamed at a later point in the overall processing pipeline, which retains the same sequencing as the original DOVER algorithm. This process happens in the same segment outlined in Algorithm 1, however is omitted for simplicity.

In the case of a speaker lock, the duration of regions is calculated as usual. In the centre column of Figure 4.2, it is illustrated—with no duration locks involved—that the consensus region is resolved to the start time of speakers B and Y and end time of speakers J and Y, due to majority voting. Due to the audio of *ex2-in2.rttm* being manually checked and speaker J being locked, this is the speaker label chosen in the output. The leftmost column contains no speaker lock, therefore, the first input's label, in this case speaker A, is

Algorithm 1 Flagging overlapping regions based on lock state

```
overlaps ← OpenOutputFile()
hashmap ← CreateEmptyHashmap()
while not end of file1 do
    region ← ReadNextRegion(file1)
    if region.type == "SPEAKER" then
        hashmap.append(region.start, region.duration, region.speaker,
            region.duration_lock)
    end if
end while
while not end of file2 do
    region ← ReadNextRegion(file2)
    if region.type == "SPEAKER" then
        for item in hashmap do
            if OverlapExists(item, region.start, region.duration) then
                if item.dur_locked then
                    overlaps.print(file2, region.start, region.duration)
                else if region.dur_locked then
                    overlaps.print(file1, item.start, item.duration)
                end if
            end if
            // speaker lock and conflict checks
        end for
    end if
end while
```

chosen in the output. The rightmost column contains a region which has both a duration and speaker lock, preserving both region bounds and speaker label in the output.

Both Locked

Regions can be assigned both the duration and speaker lock attributes. This represents cases where the user has confirmed the edit of a speaker name and region start/stop times. The logic implemented to retain the start/stop times in duration-locked regions and speaker name in speaker-locked regions is designed to allow the preservation of both properties in "both locked" regions.

4.2.4 Lock Conflicts

Following the editing lifecycle of an audio file in a digital collection, its diarisation data may be corrected and combined using the lock-aware DOVER such that there may eventually be lock conflicts between inputs. That is, a locked region from one input may overlap a locked region from another input. This occurrence could be manifested by two editors disagreeing about a speaker name, for instance, or by a mistake made by an editor who locked a region that held incorrect properties.

There are numerous potential combinations of conflicts involving duration, speaker, and both locks. Due to the nature of the diarisation editor in Greenstone (detailed in Chapter 5), the maximum number of conflicting regions that can occur is two. Figure 4.3 depicts a duration/speaker, duration/both, and speaker/speaker conflict respectively in each of the three loosely apparent columns.

Regions involved in such conflicts should be merged in a helpful way that allows an editor to easily identify and resolve the conflict. This is achieved in two stages: one for duration conflicts and another for speaker conflicts.

To help conceptualise the differences between the output generated by DOVER and DOVER-Lock, Figure 4.4 shows the output when DOVER is

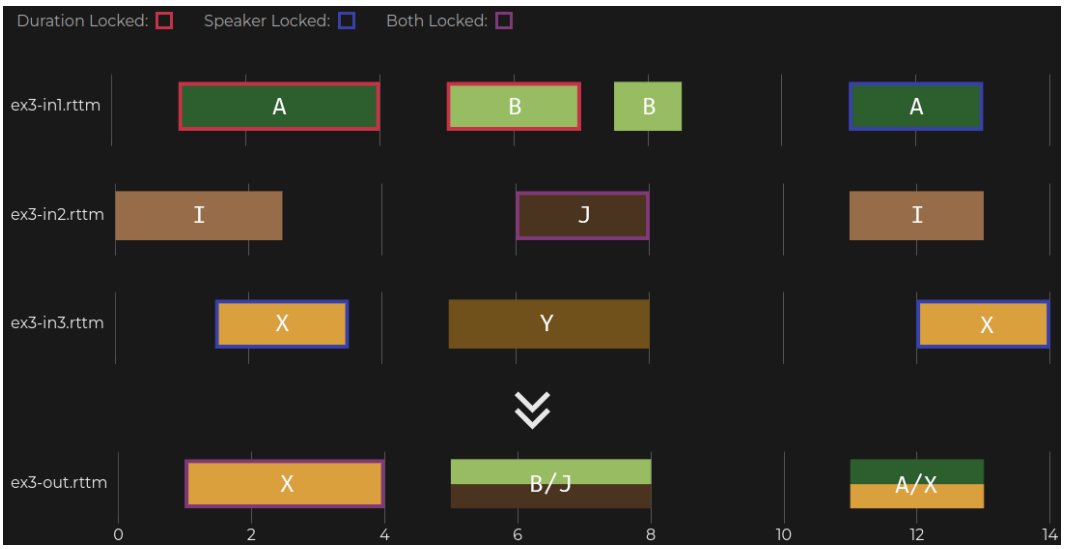

Figure 4.3: Region Lock Conflicts Example (*Note:* In the output RTTM, regions with two colours represent speaker conflicts involving two different speaker names.)

Figure 4.4: Diarisation Output Combination without Region Locking

run with the inputs shown in Figure 4.3. Figure 4.5 shows the output when executed using DOVER-Lock. Respecting the user edits denoting which attributes should be retained for each region results in a more accurate and informative outcome.

Duration Conflicts

A duration conflict asserts the disagreement of region start and/or stop times between two input files. If two regions overlap and are both duration-locked, the output region is represented by the minimum start time value, maximum end time value, and a speaker name with the format *dur_lock:conflict:speaker_-01/speaker_02*. Prepending "dur_lock:conflict:" ensures the editor or viewer can readily identify that the region has unresolved duration conflicts. Including both speaker names establishes the two sides of the conflict, giving the editor

Figure 4.5: Diarisation Output Combination with Region Locking

pathways for resolution.

Speaker Conflicts

If two regions overlap and are both speaker-locked, the script renames the consensus region to contain both conflicting speaker names, such as *spkr_-lock:conflict:speaker_01/speaker_02*. This conflict asserts that there has been a disagreement or error in the assignment of speaker names to a particular region between input files. The retention of conflict types is discussed in Section 4.2.5. The earlier Figure 4.3 shows how speaker A and speaker X in the third column of regions are equally represented in the consensus output.

Both Locked Conflicts

In cases where two regions overlap and both contain duration and speaker locks, the minimum start time and maximum end time is extracted and deemed the consensus region. The region is renamed in the manner mentioned in the previous cases, ensuring both conflicting regions are represented in the output.

There are many crossover cases which adhere to the following rules:

- If two regions with duration locks are compared, the union of both regions is taken.
- If two duration-locked or speaker-locked regions are compared, the speaker name is changed to reflect both conflicting parties.
- All conflict cases result in *conflict:* being prepended to the consensus region.

4.2.5 Output Cleaning

The original output of DOVER does not support region locking. This, along with speaker names not being properly formatted, requires additional steps to be taken to restore the output to a more accurate representation of the combination.

Name Restoration

To aid with label mapping, DOVER prepends the corresponding input number to each speaker name. Speaker names are also converted to lowercase as *md-eval.pl* requires lowercase speaker name inputs. For example, a region from the second input RTTM with an original speaker name of "Jim" will be renamed to "sys2:jim". The speaker names in the output are not reverted in the DOVER script, which results in pre-text that is useless to the user and the loss of casing. To rectify this, a method was developed to restore the original speaker names.

As the input files are first being processed, a file named *unique_speakers* is generated which contains all speaker names across all inputs with their original casing. The following represents a simplified version of the renaming process where *unique_speakers* is checked for matches:

```
function rename_to_orig(input_name) {
  while (getline orig_name < unique_speakers > 0) {
    // removes 'sys:x'
    gsub(/sys[0-9]:/, "", input_name);
    // restores casing
    gsub(tolower(orig_name), orig_name, input_name);
  }
  close(unique_speakers);
  return input_name;
}
```

The function *gsub*, provided by AWK, is a string replacement function with

parameters for *matching regexp, replacement string*, and *target* respectively, with the substitution applied globally to the whole string.

Locked Outputs

A function was created so that the consensus output of the lock-aware DOVER script retains duration or speaker lock states based on various conditions. After diarisation outputs are processed through the lock-aware DOVER script, regions that were once locked should still retain their previous lock state(s). The preservation of user edits is critical and to reflect this, the original lock states are retained so long as some conditions are met. Firstly, the lock state should not be retained if there is a conflict involving that specific lock type. For example, if a duration-locked region overlaps with a region that is both duration and speaker-locked, the output would only preserve the speaker lock.

Conflict Type Prepend

To better represent and visualise regions in the enriched Greenstone audio player, the type of conflict(s) detected is prepended to the speaker name. This ensures users or editors know the origin of the conflict and more specifically whether a given region represents a speaker or duration lock conflict, or both.

4.2.6 Summary

Figures 4.1–4.3 illustrate examples where no locks, locks, and conflicting locks exist respectively. In all cases, the output is generated to best represent the inputs so that further viewing and editing is intuitive.

4.3 Discussion

The work completed to add the support of region locking in diarisation outputs to retain user-edited data throughout output combination saves high-quality manual edits. In the lifecycle of audio within a digital collection, improvements

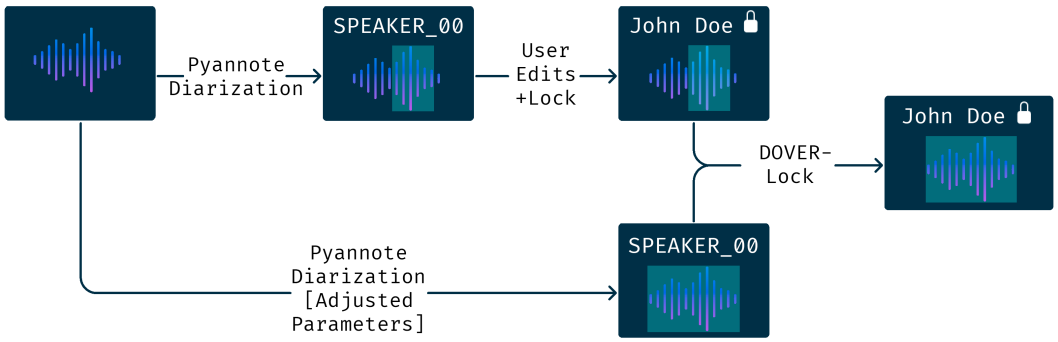

Figure 4.6: Audio Diarisation Output Combination Pipeline (*Note:* The best attributes from both streams are combined by using targeted locking in DOVER's merging algorithm.)

may be made to either the audio—for example, noise reduction—or to the diarisation system. To benefit from these improvements as well as the existing user edits, the DOVER-Lock script allows for the best of both inputs to be extracted. Figure 4.6 demonstrates a case where Pyannote parameters were adjusted to result in a more accurate diarisation output. Here, the adjusted parameters step could instead represent the same diarisation system being applied to audio that has had a noise reduction filter applied to it, or instead represent a completely different diarisation tool.

As outlined in Chapter 3, human-in-the-loop machine learning is a set of strategies for combining human and machine intelligence in applications that use AI (Monarch, 2021). Typical HITL systems utilise user corrections in an attempt to improve following predictions. While useful in cases where input data is abundant, in smaller-scale digital collections where user edits are highly valuable, there is a chance that edits will be lost throughout a re-run of the content analysis system. This would manifest itself as the rebuilt digital collection having the same error in it again, despite it being previously manually corrected.

4.3.1 OCR

The concepts implemented to preserve regions that have been locked are transferrable to systems using OCR. Locked sections of text could retain their contents in the case where outputs are combined, or the OCR tool is re-run. This would ensure the preservation of user edits while non-locked sections gain the benefit of being merged or reprocessed.

A key difference between combining OCR and diarisation outputs is that diarisation outputs generally do not need spatial alignment. That is, it is expected that the inputs are derived from the same audio source. The string alignment methods mentioned in Chapter 3 mitigate the OCR output alignment problem. Consensus voting methods outlined in both Ocromore and the paper by Wemhoener et al. exercise majority voting. By overruling the majority voting by first checking if a word is locked, giving precedence if it is, a segment-locking system could be developed to achieve a result similar to the improved DOVER script.

4.3.2 Limitations

While this thesis aims to enhance the capabilities of the Greenstone 3 Digital Library and the DOVER script, it is essential to acknowledge certain limitations. Firstly, the improved script cannot handle cases of region overlaps in the output RTTM. This limitation is beyond the scope of the current project and would require further research and development. Built into DOVER-Lap is the ability to handle overlapping segments, however, as mentioned previously, it is not suitable for combining only two RTTM files, which is the most common scenario within a digital library which supports the combination of manual corrections with a re-run of a content analysis algorithm. The diarisation editor and utility scripts developed to bridge interactions between the DOVER-Lock script and Greenstone interface both support overlapping regions. However, because the methods used to give precedence to locked regions rely on overlaps, the DOVER-Lock script would require significant adaptation to support

this feature.

4.4 Summary

This chapter has presented DOVER-Lock, an extension of the existing DOVER algorithm that allows specific segments of inputs to be retained in the combination process. This was achieved by introducing two types of region locks and further enriching the functionality by integrating conflict resolution steps. This chapter has also mapped out, in general terms, how the technique could be applied to other forms of content analysis such as OCR or ASR.

The ability to lock regions that have been manually corrected ensures edits are not overwritten, preserving the valuable time and effort offered by the editors. Providing the options of locking either the start/stop times of a region, the speaker name, or both, gives the DOVER-Lock algorithm room to generate a high-quality output. The retention of lock and conflict types results in further edits being more approachable, improving the edit lifecycle of diarisation data.

The analysis of the DOVER script and its merging strategies helped develop an understanding of the shortcomings involved in the loss of user edits. This allowed for the implementation of DOVER-Lock, rectifying user edit retention and providing further improvements such as lock preservation and conflict handling.

Chapter 5

Diarisation Editor

5.1 Introduction

In this chapter we describe the work undertaken to enhance the capability of the Greenstone 3 Digital Library system so that it can support the iterative combination of audio-based content analysis for diarisation and manual corrections. By default, when supplied with an audio document, Greenstone 3 shows an HTML5 audio player. To enhance the audio interaction, through a previous project, the author undertook the development of an enriched audio player which provides a more in-depth interface that also incorporates diarisation visualisation. In this chapter, we detail the work necessary to develop this into a fully functional audio diarisation editor that is embedded into the digital library interface. The work is available through the primary Greenstone 3 Digital Library codebase.

We start this chapter with a description of the existing audio visualiser interface to give a thorough understanding of the state the audio player was in before the thesis commenced. The JavaScript library powering the audio player will then be outlined before the mechanism of Greenstone's file-level document version history is explained. This is a new general capability added to the Greenstone software architecture by the digital library programmers who work on the project. The addition was prompted by the work reported

in this thesis, however, its applicability goes beyond the scope of this work. Lastly, Section 5.4 explores the developments made to expand the existing audio player into a fully functional diarisation editor.

The editor is an opt-in feature in the Greenstone 3 software, controlled through a collection's configuration file. The audio is processed through Pyannote, generating "who speaks when" data (on this occasion in CSV format, but comparable to the RTTM format seen in Chapter 4) and attaching it to the corresponding document. When viewed, the default HTML5 audio player is instead replaced with an enriched audio player showing speaker regions on a waveform.

5.2 Existing Interface

Before the diarisation editor improvement, the interface Greenstone 3 provided, when displaying structured audio in its document view, was comprised of a waveform, a toolbar, and a chapter list. The waveform represents the audio data and scrolls as it is being played. Speaker segments can be viewed on both the waveform and chapter list, which is expandable by clicking the chapter icon at the left side of the toolbar. Speaker segments can be played by clicking either the region on the waveform or the chapter in the chapter list.

Basic controls are available on the toolbar including a waveform zoom slider, play/pause button, skip forward and back buttons, and a mute button. These buttons are found on popular media players including YouTube and Spotify, which makes their usage intuitive. Figure 2.12 on page 33 shows the state of the audio player before any additions were made as part of this thesis.

5.2.1 WaveSurfer.js

As briefly introduced in Section 2.3.2, Wavesurfer is a web browser-based JavaScript library that provides customisable audio waveform visualisation. It operates on all modern browsers supporting Web Audio including Firefox,

Chrome, Opera and Safari. It offers numerous plugins and functions to enrich interactions with audio, which were used to develop the digital library's structured audio player document view. Wavesurfer provides a plugin named *Regions* which closely correlates to diarisation segments.

Regions are visual overlays on a waveform that can be used to encapsulate sections of audio in a stylised area. Regions in Wavesurfer are generated with a start time, end time and a colour which can be used to represent speaker identity. Speaker regions on the waveform are coloured based on the speaker name, which helps differentiate between different speakers.

The region colours used by Wavesurfer.js are customisable. For use in Greenstone 3, the range of colours used were generated with *Colorbrewer*[1] which is a tool designed to help mapmakers select effective colour schemes for thematic maps. When given an integer, *Colorbrewer* provides the specified number of colours, with each colour being easily separable from the other. This ensures similar colours are avoided, preventing any confusion distinguishing speakers.

5.3 Design

The objective of the editor design was to expand on the existing audio player to allow for the editing of diarisation data. Developments were made to allow manipulations between two diarisation output versions. The editing interface was further enriched to provide intuitive interactions. A means to combine two diarisation outputs using the previously described DOVER-Lock algorithm was also implemented.

The diarisation editor web application was developed using JavaScript, XSL, and CSS utilising the JQuery, Wavesurfer, and Colorbrewer JavaScript libraries. This was combined with Greenstone 3's document editing API and document ingest process to support the required diarisation lifecycle.

[1] https://colorbrewer2.org

5.3.1 Architecture

The architecture of the diarisation editor centres on the configuration of a collection that conforms to a particular set of criteria and a custom-built web tool. Specifying the collection to use the *StructuredAudioPlugin* with the *segment_audio* plugin option provided means an initial version of the diarisation data will be generated when the collection is built for the first time. The custom-built web tool includes numerous core objects which reflect the diarisation data in a form that can be actively manipulated in real-time as edits are made. In this section, both the structure of Greenstone 3 and the web tool are described.

Greenstone 3

To provide support for document versioning and speaker searching, various improvements were made to the Greenstone architecture by the Greenstone development team. The intricacies of the Greenstone architecture and design are outlined in Section 2.1.2.

Prior to this thesis, Greenstone provided a general ability to edit document content and metadata by making changes and rebuilding the collection. However, if using this feature in the digital library, then—except in a few special cases—this means that the *import.pl* process can no longer be run across all documents within a collection. New documents can be added to import and an incremental-add version of *import.pl* could be run, but a full rebuild would cause the manual edits, which are stored in the archives directory, to be erased. Introducing the concept of file-level document version history to the ingest process provides a way to keep track of information (such as metadata edits) that would otherwise be lost. To allow for multiple versions of the same document, a file structure was devised and developed by the Greenstone development team to contain each version. The following outlines the new directory structure employed when more than one version of a document exists:

```
archives/[documentid]/
├── Audio.mp3
├── doc.xml
├── diarisation-data.csv
└── FLDV_History/
    ├── nminus-1/
    │   ├── Audio.mp3
    │   ├── doc.xml
    │   └── diarisation-data.csv
    └── nminus-2/
        └── ...
```

The most recent version's files exist in the document's root directory. Subsequent versions exist within the *FLDV_History* (file-level document version history) directory, inside directories named *nminus-x*, where x is the version number, incrementing from one. In the above example, *nminus-1* represents the second most recent version and *nminus-2* represents the third. Corresponding runtime API access functions were integrated so that a specific version of the document can be retrieved—this is how previous versions of audio and diarisation data is retrieved for use in the diarisation editor interface.

5.4 Diarisation Editor

The diarisation editor has been built to support multiple concurrent audio versions along with corresponding diarisation data. As will be seen below, the principle way the editor is designed to be used is to show two versions of the processed data side-by-side for the purpose of comparison: one audio track with diarisation segments, for instance, being the most recent version that has been manually corrected, and the other being the result of reprocessing using content analysis. To cater for this, an object named *SpeakerSet* was defined to hold integral information. Each version of an audio document has

a *SpeakerSet* object to contain useful data. The fields in the object include *isSecondary*, *uniqueSpeakers*, and *speakerObjects*.

The *isSecondary* property is a Boolean value and signifies whether the audio is a primary or secondary track used in dual mode when two tracks are being shown at once. The *uniqueSpeakers* property is a list of all unique speaker names that appear in the diarisation data. When colouring the regions, this property is utilised to generate a distinguishing colour for each unique speaker. It is also set as document metadata within Greenstone so that speaker names appear in collection search results.

To represent the diarisation data in a more succinct form than that found in the document's associated CSV file produced by Pyannote, the *speakerObjects* property is used. It contains an array of objects, one object for each region. Each object contains the following properties:

- **start**: The start time of the region in seconds [*Float*].
- **end**: The end time of the region in seconds [*Float*].
- **speaker**: The speaker name assigned to the region [*String*].
- **locked**: Whether the region has been locked by the user or not [*Boolean*].
- **region**: Wavesurfer region object reference [*Object*].

When a Wavesurfer region is created, an object to represent it is built. Along with the basic start and end time properties and associated HTML element, several methods are attached to get and set its properties. These methods allow for regions to be edited in place without completely reloading the entire set of regions.

An example SpeakerSet assigned to an audio document version may take the following structure:

```
{
  isSecondary: false,
  uniqueSpeakers: ["James", "Jim", "John"],
  speakerObjects: {
```

```
    [ start: 4.2, end: 7.8, speaker: "James",
      locked: true, region: {...} ],
    [ start: 10.3, end: 16.0, speaker: "Jim",
      locked: false, region: {...} ],
    ...
  }
}
```

The *SpeakerSet* corresponding to the currently open audio document is populated when the document is loaded. All properties are updated as start times, end times, lock states, and unique speakers are changed in the editor interface. This allows for a concurrent version of the diarisation data, making manipulations and state saving straightforward. The ability to save the state at any point is a crucial feature to allow for undo and redo operations as well as being able to save the current version at any point in time.

5.4.1 User Interface

Outlined here is the user interface of the developed audio player and editor used to show an audio document in a Greenstone collection. The objectives of the user interface design were to provide access to all of the familiar audio interactions as well as give the ability to view, navigate, and edit the diarisation data generated by the Greenstone ingest process. The layout has minimisable components such that it may be used by those wanting to only interact with the audio without bloating their experience with editor functions.

Overview

The audio interface is comprised of a waveform display, toolbar, chapter list, and edit panel. Figure 5.1 shows the interface when it is initially loaded with no editing permissions. In this state, the interface is geared to the user playing, rather than editing, content, and in appearance is similar to the player-only interface that was developed in prior work. The waveform displays the currently

Figure 5.1: Improved Audio Player in Greenstone Document View

loaded audio with timestamps on the lower border. Regions are represented on the waveform by coloured sections encompassing the time boundaries of corresponding speaker segments. The toolbar contains various buttons for audio navigation and control. The chapter list displays all speaker segments contained within the open audio item. Region properties are managed in the edit panel, which can be seen in Figure 5.2 in the lower-right section. As previously outlined, components such as the chapter list and edit panel may be minimised should the user not wish to interact with them, allowing a modular interface that may be adjusted to the desired complexity.

When in edit mode, regions can be selected by clicking within their bounds on the waveform or by clicking on the chapter list. Multiple regions can be selected by holding the shift or control keys while clicking. Regions are highlighted in red to signify they are selected. A region must be selected to edit its properties. If the "Select All" checkbox is clicked, all regions with the speaker name inside the speaker text box are automatically selected. The region selection characteristics are synonymous with file explorer interface interactions in popular operating systems such as GNU/Linux and Windows.

To be able to access these features, the user accessing the Greenstone collection must be logged in to an account that has been granted sufficient privileges

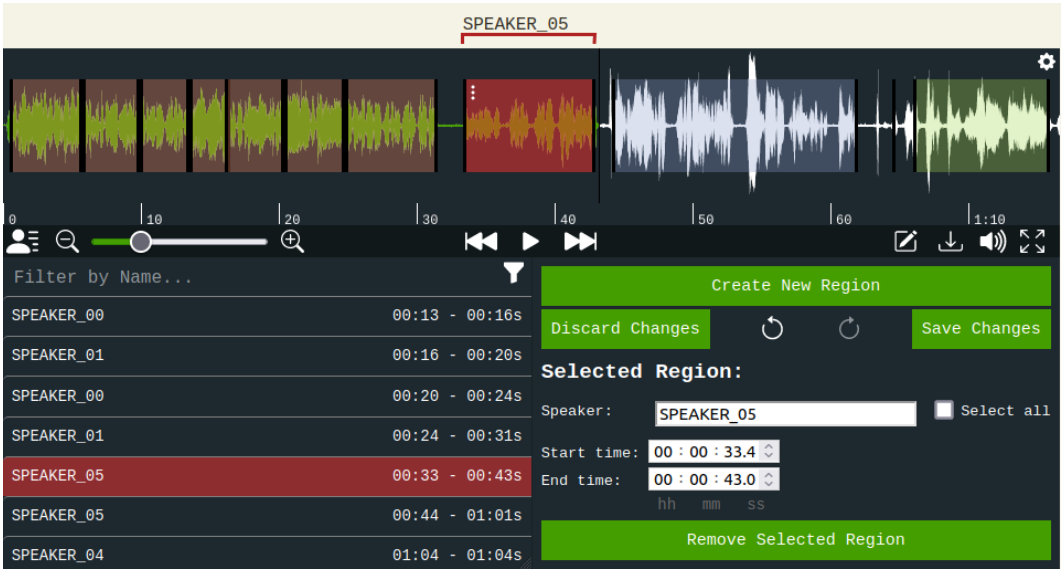

Figure 5.2: Improved Audio Player in Greenstone Document View with Selected Region (*Note:* The selected region is indicated by the boundary bar above and red background on both the region and chapter.)

to edit the collection. If the required privileges are not held or edit mode is disabled, audio can be played, and regions can be viewed on the waveform however all editing actions are made unavailable. Employing a security measure to restrict edit access, as highlighted in Section 3.1.2, is a fundamental aspect of the design so that unauthorised users are unable to manipulate potentially significant region data.

Figure 5.2 shows a region with the speaker name "SPEAKER_05" as the selected region. Also shown is the audio timeline, toolbar, chapter list, and edit panel.

Figure 5.3 shows (from left to right) the version menu, region menu, and timeline menu expanded. All of the expanded menus, aside from the *Hide Regions* option in the timeline menu, are made inaccessible if the user either lacks edit permissions to the collection or is logged in with an account that has sufficient permissions however has not enabled editor mode in Greenstone.

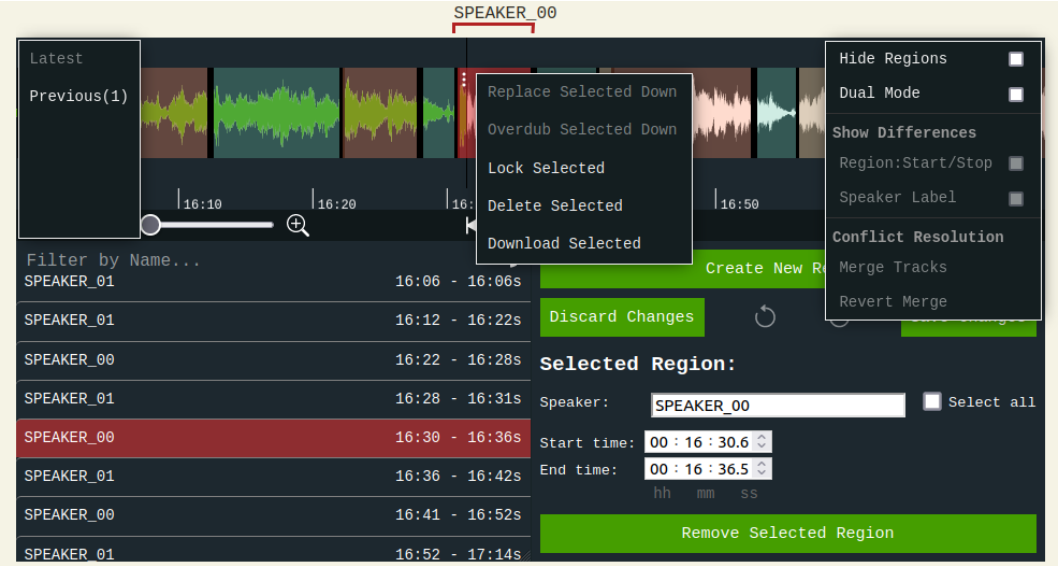

Figure 5.3: Improved Audio Player with Expanded Menus (*Note:* There are several options in the menus that have been disabled due to various requirements, primarily the need for the editor to be in dual mode.)

Creation and Deletion

Regions are created and deleted by clicking the *Create New Region* and *Remove Selected Region* buttons respectively. When a region is created, it appears on the chapter list and waveform at the current play cursor location with a placeholder name "*NEW_SPEAKER*". The deletion process simply removes the selected region or regions from the chapter list and waveform.

Editing

Regions possess three primary properties: a start time, end time, and speaker name. Further properties include lock and conflict state. Start and end times can be edited by click-dragging one of the region bounds or by using the HTML5 number input to more precisely set a desired value of the selected region. If same-speaker regions are adjusted so that they overlap, the regions are merged due to overlapping same-speaker regions being illogical. The editor supports different speaker region overlaps.

The speaker name can be edited by typing in the textbox with the label

Figure 5.4: Locked Region (*Note:* Regions can be unlocked by accessing the region menu or by clicking the lock icon.)

"Speaker:". This instantly updates the text in the chapter list and above the waveform as text is entered. The edit panel is visible in the lower right section of Figure 5.2.

To signify that a region has been edited and/or confirmed, a lock can be applied by accessing the region menu. Multiple regions can be locked or unlocked at once by selecting multiple regions and then toggling the lock state. The current lock state of a region can be identified by the presence or absence of a padlock symbol as shown in Figure 5.4. The padlock can also be seen in the chapter list.

Undo, Redo, and Saving

A user's interaction session with this interface, due to its intricate nature, is likely to be a complex and time-consuming task. For this reason, it was important to implement a robust undo/redo capability so that edits can be made with confidence. A simple model commonly known as a linear undo model is implemented. This model is widely used, and its simplicity is a big advantage as the result of an undo/redo operation may be easily predicted by the user (Jakubec et al., 2014).

For each editing action the user makes, the current state is saved to an undo state stack. An undo stack pointer is incremented and decremented as the undo and redo buttons are clicked, as well as the undo state item at the index of the stack pointer being loaded into the current view. To compliment

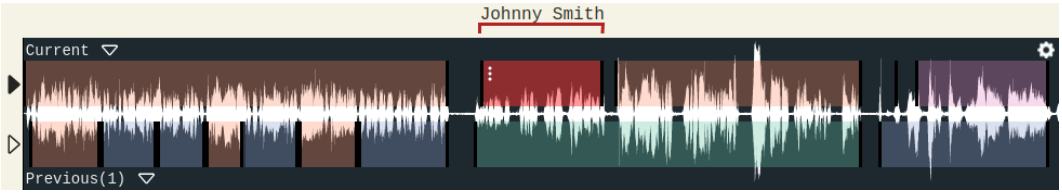

Figure 5.5: Dual Mode Timeline View

the undo and redo functions, the browser LocalStorage component was used to store the current state of the editor and undo history. In the case the browser crashes, or for other reasons is closed, the user can not only return to editing the document and find their editing work has been preserved, but also their undo/redo history has been reconstituted.

Diarisation edits may be saved, in the form of a new version, back to Greenstone's file-level document version history by clicking the *Save Changes* button. This prompts the user to enter a commit message, which when submitted, attaches to the new document version as metadata. This commit message can be seen in the interface when hovering items in the version list. The *Discard Changes* button reverts the diarisation data state back to its original form before any edits were made to that version.

Dual Mode

Dual mode is an option made available when two or more versions of a particular audio document exist in Greenstone's file-level document version history. It is a toggleable option and when enabled adapts the waveform to display two diarisation datasets. Figure 5.5 shows the dual mode view with two different diarisation datasets. Two pointer arrows are shown to the left of the waveform to signify which of the two sets is selected, which may be clicked to change the selected set. The selected set has its regions listed in the chapter list and audio shown on the waveform. The version name and dropdown menu to change the selected version(s) can also be seen on the left end of the waveform.

When dual mode is enabled, two actions are made available in the region menu, *replace* and *overdub*. These actions apply to the opposite set. *Replace*

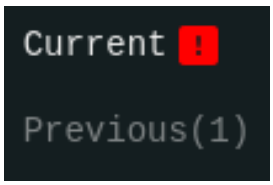

Figure 5.6: Version Select Menu Item Containing Conflicts

copies the selected region or regions to the opposite set, removing any regions that overlap with the new region or regions. The *overdub* also copies the selected region or regions to the opposite set, however, will merge any same-speaker overlaps instead of deleting them.

If a region is resized using the drag handles, the handle will snap to start and end times in the opposing set. This helps align regions between the sets. If required, for precise adjustments, the snapping feature can be temporarily disabled by holding the control key while resizing or by changing the start/end time in the edit panel.

Lock Conflicts

As outlined in more detail in Section 5.4.2, DOVER-Lock can be instigated through the interface, merging both visible diarisation sets. As a result of a DOVER-Lock merge, there may exist conflicts in the new diarisation data. In this context, a lock conflict exists when two regions with the same type of lock have been merged. It is important to highlight lock conflicts because two regions with the same lock cannot be merged with a clean resolution. Versions containing lock conflicts have a red exclamation mark next to the version name as shown in Figure 5.6.

As outlined in Chapter 4, the two types of lock conflicts are speaker and duration. These are prepended to the speaker name as "spkr_lock:" and "dur_-lock:". If a region's speaker name contains one of those strings, the string is replaced with a corresponding icon. Figure 5.7 shows the title given to a region with the speaker name "spkr_lock:dur_lock:John/Jim". When hovered, the icons show a tooltip to assert what the icon is representing.

Figure 5.7: Region with Lock Conflicts (*Note:* The conflict flag icons are removed when the editor removes the corresponding text from the region speaker name.)

In practice, when using the full set of features provided by the editor, the user may resolve a conflict. This could involve simply confirming a conflict between two speaker names, however, could also entail multiple speaker and start/end time conflict resolutions. Using the *Save Changes* button, as mentioned previously, saves the changes back to the collection in the form of a new version.

Searching

Searching has two facets in the diarisation-supporting digital library realm. The first involves searching for speakers across a collection containing multiple audio documents. This is achieved by utilising the aforementioned *uniqueSpeakers* attribute attached as metadata to each document. To ensure results remain up to date, the metadata is generated using a Perl script when the collection is built, and is updated using JavaScript should the user save any speaker name changes. Greenstone provides methods for both Perl and JavaScript that allow for metadata to be manipulated on live collections. Figure 5.8 shows the Greenstone search results when queried with the string "John".

The second facet of searching is made available in the audio player's interface at the top of the chapter list. As the user types in the text box, the listed regions are filtered based on speaker name matches. The non-matching regions are omitted from both the chapter list and waveform. Figure 5.9 displays the

Figure 5.8: Collection-Wide Searching for Speaker Name in Greenstone

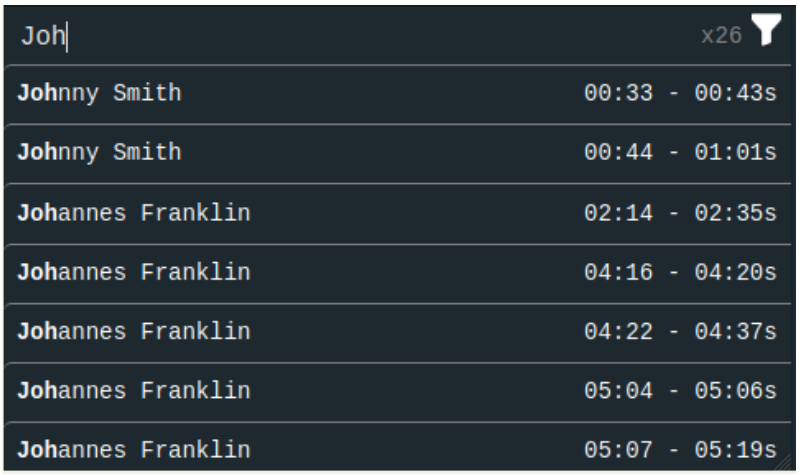

Figure 5.9: Searching in the Chapter List (*Note:* The number of matching results can be seen on the right-hand side of the search box.)

search function in use along with the number of matching regions on the right end of the search box. This helps users access all available speaker segments with matching speaker names.

Additionally, users can filter the regions by clicking the filter icon on the right-hand side of the search box and adjusting the minimum and maximum duration sliders. This updates the chapter list and visible regions on the waveform in real-time as the sliders are adjusted. The functionality is useful for locating very small or very large audio segments. Figure 5.10 shows the minimum and maximum sliders returning regions between twelve and fourteen seconds in duration. The duration filter box can then be closed by clicking the filter icon once again.

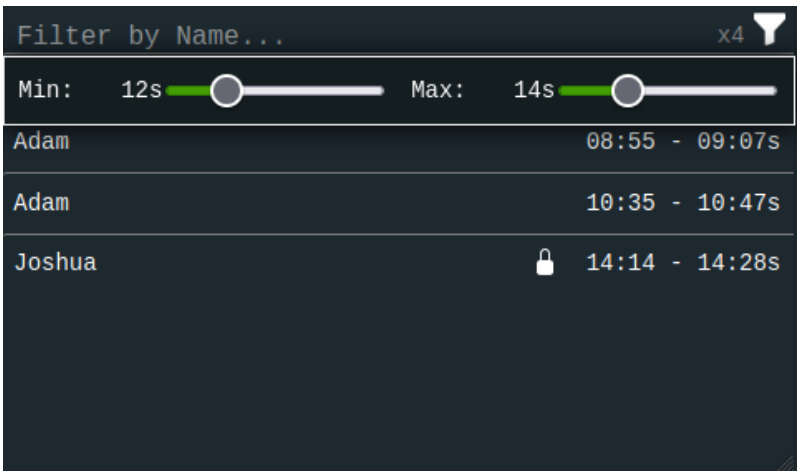

Figure 5.10: Duration Filter in the Chapter List

Extra Improvements

To further extend the proficiency in the diarisation editor, several additions were made to enrich the editing experience and capability.

- The entire audio can be downloaded as an MP3 file from the toolbar. Region audio can also be downloaded by accessing the region menu either by right clicking a region on the waveform or clicking the three vertical dots on a selected region.
- A fullscreen mode was implemented to make editing easier. As well as hiding the surrounding Greenstone interface, the waveform is enlarged vertically to make working with two tracks more user-friendly.
- From the timeline menu, there is an option to hide all regions. This also collapses the chapter list and editing panel to strip all diarisation-related elements from the interface.
- Tooltips are present on all interactable items on the interface, giving a short description of what the hovered item does.

5.4.2 Instigating DOVER-Lock from Greenstone

To bridge the connection between the DOVER-Lock script and the interface just outlined, the diarisation editor supplies options for initiating the DOVER-Lock merge. Firstly, the objects present in the editor need to be converted to the appropriate RTTM file. The process can be instantiated through the user interface which generates and displays the consensus output. This process creates a new document version which is appended to the document history folder structure.

Lock Assignment

The padlock locking ability in the diarisation editor indicates a duration lock in terms of DOVER-Lock. The speaker lock property is set automatically if a speaker name has been changed from the Pyannote-generated "*SPEAKER_-XX*". As outlined in Chapter 4, the updated RTTM file format devised for DOVER-Lock reflects the lock states by including three more fields in the tenth, eleventh, and twelfth columns. These fields are binary entries representing duration lock, speaker lock, and a lock that is true (1) if either of the previous locks are true. The latter is an attribute included in the RTTM files that makes areas of the scripting easier to express.

The automatic assignment of the speaker lock was selected as offering two different locking methods in the interface may confuse the user. Furthermore, it is reasonable to assume that a change to a speaker name indicates the user has made an educated edit. Because the output can be compared to any other version, any unexpected speaker names can be identified using the aforementioned conflict flag (see Figure 5.7) and corrected in the dual-mode view.

Actuating DOVER-Lock

Before DOVER-Lock can be called, the *SpeakerSet*s in the JavaScript code need to be converted to RTTM files to be used as input to DOVER-Lock.

Then, in the interface, the DOVER-Lock process can be initiated by firstly viewing both tracks to be merged in the dual-mode view. Clicking the *Merge Tracks* option in the timeline menu will initiate the DOVER-Lock process.

The two RTTM files are passed to the DOVER-Lock script, which generates a new RTTM file which is parsed into a SpeakerSet to be shown in the diarisation editor. Any conflicts encountered in the DOVER-Lock process are reflected in the interface.

The connection between Greenstone and the DOVER-Lock script is bridged using Perl CGI scripts structured as modules that define actions to interact with the Greenstone file system. These actions are accessible through AJAX calls when given the appropriate site, collection, and document parameters pertaining to the target resource. To satisfy the capability of Greenstone being able to run DOVER-Lock, a new action was added to the Perl CGI scripts, responding to a POST request, which initiates the execution of DOVER-Lock using the input arguments provided as the RTTM input files and returning the result as its response to the POST request. Because DOVER-Lock operates using input/output files, utility scripts were created to convert between RTTM strings and files. The following outlines a simplified version of the DOVER-Lock actuation function, where *a* represents the desired action and *inputitems* is an array of RTTM strings:

```
const actuateDoverLock = () => {
  let primaryRTTM = speakerSetToRTTM(primarySet);
  let secondaryRTTM = speakerSetToRTTM(secondarySet);
  $.ajax({
    type: "POST",
    url: doverServerURL,
    data: {
      "a": "run-dover",
      "inputitems": [primaryRTTM, secondaryRTTM]
    }
```

```
    }).then(dover_output => {
      // update new primarySet
      // update unique speaker list for gs3 search index
      // update associated file
      // disengage dual mode
    });
  }
```

By default, *primaryRTTM* and *secondaryRTTM* represent the top and bottom speaker sets respectively.

Back in the Greenstone interface, using JavaScript, a new document version is created, its associated file is set to the new diarisation data, and dual mode is disengaged so that only the DOVER-Lock output is displayed to the user. The previous versions are still accessible in the version dropdown menu. The document version incrementation and associated file assignment are both actions contained in a Perl module similar to the one used for DOVER-Lock, allowing AJAX requests to be used to manipulate the Greenstone collections.

DOVER-Lock Example

The following demonstrates the process of merging two versions of audio documents which have had a diarisation algorithm applied from the user interface within a Greenstone collection. Figure 5.11 displays two speaker sets in the dual mode view. The primary (top) set differs from the secondary (bottom) set as the first three regions have been renamed from *Alex* to *Jeffery*. The second region of the bottom set has been split to better represent the speaker segments. In this example, all visible regions in both sets have a lock applied.

The DOVER-Lock merge is then applied, generating the output shown in Figure 5.12. Because both regions have duration and speaker locks attached, the resulting regions contain duration and speaker lock conflicts. The speaker names reflect that both *Jeffery* and *Alex* are speaker-locked regions. With this conflict visually displayed, continuing the example, the user would use

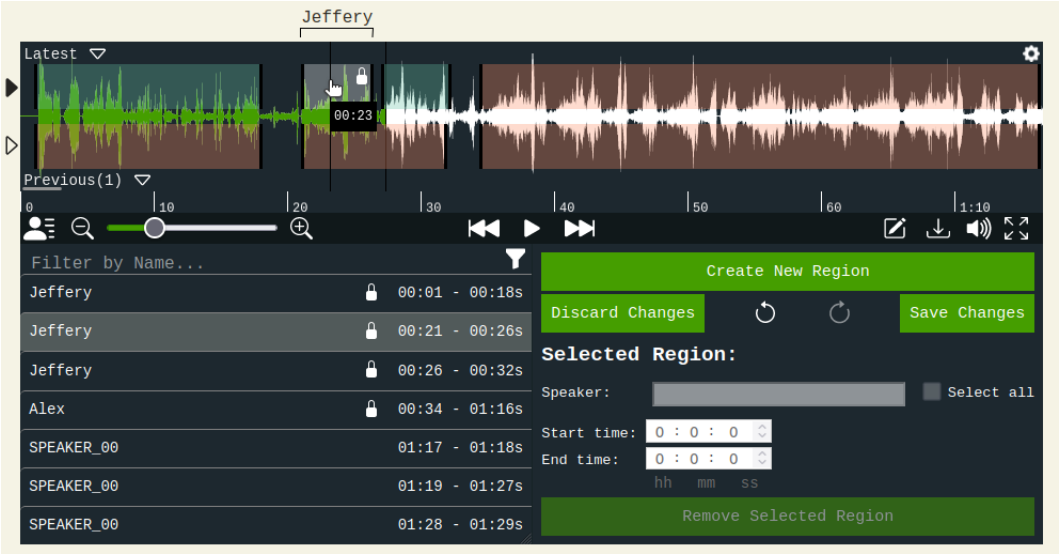

Figure 5.11: Two Sets Prior to DOVER-Lock Merge (*Note:* The secondary (bottom) set contains locked regions with the speaker name 'Alex'.)

the play features of the editor to listen to that segment, and upon reaching a determination, update the speaker information then save/commit the change.

5.5 Discussion

Bringing versioning to Greenstone is a valuable feature of the digital library toolset. Not only does it allow digital librarians to view and revert to previous versions of a document, but it also shows the evolution of the data. Different perspectives, for example, the understandings of different iwi, may be captured in the versions. In the case of diarisation data, one editor may believe a speaker to have a particular name, but another editor knows them by another. Similar phenomenon may arise in OCR or ASR, where different opinions or perspectives could be realised by analysing version discrepancies.

The importance of versioning extends past cultural reasons, for example, to understand a tribunal sentence it is necessary to get access to the previous versions of the involved rules as they may have been different at the moment the tribunal sentence was made (Martínez-González, 2005). Digital libraries can host a plethora of different data, many of which will benefit from file and

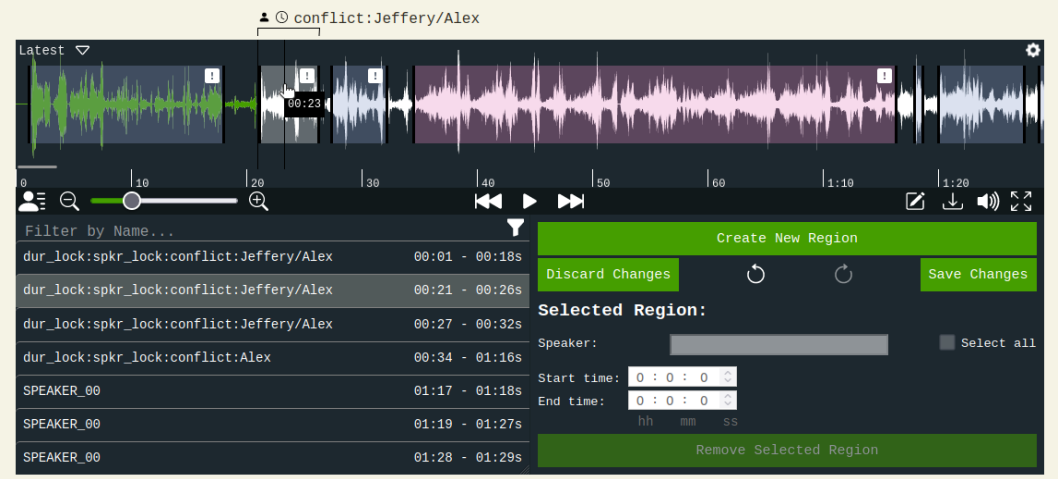

Figure 5.12: Result Of DOVER-Lock Merge (*Note:* Notice the icons to the left of the speaker name representing the speaker and duration lock conflicts found during the DOVER-Lock merge.)

metadata versions.

User testing was omitted from the thesis project due to the task-specific nature of the interface. The editor is purpose-built to cater to the specialised task of editing the output of a diarisation system. The primary user base for the editor is likely to be digital librarians managing a specialised collection of audio documents, meaning available candidates for user testing would be limited. While designing the interface, inspiration was taken from other content editors including text editors and DAWs. No existing public-domain diarisation editor was located throughout the thesis duration, meaning the majority of design decisions have been based on other audio-related interfaces.

Future work could see adequate testing of the interface to ensure users have the best chance of having a positive interaction with the tool. To help introduce new users to diarisation editing within Greenstone, documentation was developed to walk through the installation and diarisation manipulation steps (see Appendix B).

5.6 Summary

The extensions made to the existing interface have transformed the tool from a diarisation viewer to a diarisation editing suite. The ability to personify regions through renaming enriches the diarisation data, increasing the amount of information available to users. Diarisation region boundary errors can be fixed, tailoring the results to better represent the content in the audio.

Diarisation is a new addition to the digital library space, however, it provides an engaging facet of information with which users can now interact with. As transcription, OCR, diarisation, and other content analysis systems become more prevalent in modern digital technology, digital libraries should follow suit and integrate these index-improving processes.

Chapter 6

Conclusion

This thesis provides an extensive investigation into user edit preservation in content analysis output combination and the implementation of an enriched audio player and diarisation editor. The following section highlights key findings and further work that the developed systems could benefit from.

Content analysis strategies have been explored with a primary focus on diarisation data and the innovative methods available to reduce errors. The investigation led to the design and implementation of a capable diarisation editor framework within the Greenstone Digital Library environment, which not only visualises diarisation data but now empowers users to interact with and refine this data, while preserving the integrity of their corrections.

Existing diarisation output combination strategies such as DOVER and Ocromore were explored, and DOVER was adapted to support the protection of user edits as the data passes through the DOVER script. The resulting project was aptly named DOVER-Lock and provides lock conflict handling and other improvements such as speaker name restoration.

Two key findings of the work are:

- This study revealed how existing output combination techniques can be further improved by integrating locking mechanisms to preserve specific sections of data. The research builds upon the output combination schemes proposed by Fiscus, 1997, Stolcke and Yoshioka, 2019, and Raj

et al., 2021, however the concepts may be applied to similar processes for different systems such as OCR, keyword extraction, sentiment analysis, and other content analysis strategies that can benefit from user corrections.

- The edit-aware concepts developed contribute to the field of content analysis output combination and highlight the importance of preserving high-quality corrections. Particularly in a digital library context, user data is not expendable like it is in ML systems. This is due to there being a direct representation of the input data within the digital library. In contrast, ML systems have substantially more inputs and the output is not expected to directly match any of the inputs.

6.1 Future Work

Potential future work in the relevant field could include:

- Support for overlap-aware diarisation system output combination while retaining the lock-aware features developed within this thesis.
- Automatic adjustments of parameters such as minimum speaker region length based on user edit habits.
- Automatic generation of RTTM plots to aid in the visualisation of the DOVER-Lock combination process.

Additionally, Greenstone could be further developed to allow versions to be deleted and reverted to and provide a way of comparing versions of non-audio documents using a "diffing" mechanism. These items were not prioritised over the duration of this thesis as they fall outside the scope of user edit retention in output combination systems. That being said, the aforementioned improvements would help the overall system become more well-rounded and functional.

6.2 Summary

The significance of this research extends beyond the domain of diarisation itself, as the concepts and techniques developed hold applicability across various content analysis areas such as OCR and ASR. Although human-in-the-loop machine learning strategies exist, they often disregard the high-quality nature of the human edits and only *tend* to lean toward the user's suggestion. As a user in a digital library, one would expect their edits to be retained in whole and not lost in the case of a content analysis re-run or output combination.

The enriched audio player not only enhances the accessibility and usability of diarisation data but also introduces novel functionalities that facilitate the refining and editing of speaker region bounds and speaker names. Furthermore, the comparison capabilities between two diarisation outputs along with the implementation of sequence merging and region conflict resolution amplify the utility of our framework and foster a deeper understanding of previous diarisation versions and user edit history. These features enable users to explore different hypotheses, assess the quality of automated analysis, and iteratively refine the results, producing a more dynamic and accurate content analysis workflow within a digital library.

In conclusion, the findings in this thesis highlight the transformative significance of preserving human input when combining automated content analysis outputs. This paves the way for a more nuanced, accurate, and adaptive approach to interpreting multimedia data.

Appendix A

Code Repository

Primary code and utility scripts developed as part of this thesis are hosted on GitHub at https://github.com/finn-i/dover-lock. Note that not all adapted files are included here, only the most relevant. This is due to the sophisticated filesystem of Greenstone, resulting in numerous edited files which do not hold a significant number of contributions. Changes made will also be included in the latest version of Greenstone.

Appendix B

Diarisation Editor Documentation

The Diarisation Editor is a tool designed to assist in the process of diarisation, which involves segmenting and labelling audio or speech data into different speakers or conversation turns. This documentation provides an overview of the features and functionality of the Diarisation Editor, along with instructions on how to use it effectively.

Table of Contents

1. Prerequisites

The Diarisation Editor requires the following prerequisites to be fulfilled:

- A working Greenstone 3 Digital Library
- A Greenstone collection containing audio

2. Installation

To start using the Diarisation Editor, follow these steps:

1. Add:

```
<plugin name="StructuredAudioPlugin">
  <option name="-converttotype" value="mp3"/>
  <option name="-segment_audio"/>
</plugin>
```

   to ./etc/collectionConfig.xml within the *pluginList* tag.
2. Rebuild the collection.
3. Ensure the authenticated user has edit access.

3. User Interface Overview

The Diarisation Editor interface consists of the following key components:

- **Audio Waveform**: Displays the audio waveform of the loaded audio file.
- **Chapter List**: Lists every speaker segment (name and time) in the audio file.
- **Toolbar**: Contains buttons to control audio playback, including play, pause, stop, and skip functions.
- **Editor Panel**: Contains various tools and actions for segment editing, undo/redo, and saving.
- **Region Edit Menu**: Accessed by either right-clicking or clicking the three vertical dots in a region on the waveform. Contains various editing actions.
- **Audio Waveform**: Displays the audio waveform of the loaded audio file.

4. Audio Playback and Navigation

- To play the audio, click the **Play** button in the playback controls.

- Use the **Pause** button to pause and stop the audio playback.
- To seek to a specific point in the audio, click on the waveform or use the skip forward or backward buttons on the toolbar.
- Zoom in and out of the waveform using the zoom controls on the toolbar for precise navigation.
- Adjust the volume using the volume control slider found on the toolbar.

5. Diarisation Output Editing

- To edit a region, first click it either on the chapter list or waveform to select it. Multiple regions can be selected too.
- Click and drag on the waveform to define the start and end points of a segment. Start and end points can also be edited in the editor panel.
- To rename a segment, select it and type the speaker's name in the editor panel text box.
- To add a region, click the **Create New Region** button in the edit panel.
- To remove a region, first select it then click the **Remove Region** button.
- To ensure your edits are retained, lock them by clicking **Lock Selected** in the selected region menu.
- Any changes may be reverted using the undo and redo buttons.

7. Saving Data

- Save your diarisation data editing progress back to Greenstone using the Save Changes button in the edit panel. Supply a commit message to describe the nature of your changes.
- The entire audio file can be downloaded by clicking the Download Button in the toolbar.
- A specific region's audio can be downloaded by accessing the selected region menu and clicking Download Selected.

8. Shortcuts and Hotkeys

The Diarisation Editor supports several keyboard shortcuts and hotkeys to enhance productivity. Some common shortcuts include:

- **Spacebar**: Play/pause the audio.
- **Left/Right Arrow Keys**: Navigate through the audio waveform.
- **Ctrl + Z**: Undo the last action.
- **Ctrl + Y**: Redo the last undone action.
- **Delete or Backspace**: Delete the selected segment or speaker label.

Summary

The Diarisation Editor is a powerful tool that simplifies the process of segmenting and labelling audio data for diarisation purposes. By following the instructions and understanding the features outlined in this documentation, you can effectively use the Diarisation Editor to achieve accurate and efficient results in your diarisation tasks.

Bibliography

Anguera, X., Bozonnet, S., Evans, N., Fredouille, C., Friedland, G., & Vinyals, O. (2012). Speaker diarization: A review of recent research. *IEEE Transactions on audio, speech, and language processing*, *20*(2), 356–370.

Armstrong, J. S. (2001). *Combining forecasts*. Springer.

Bainbridge, D., & Witten, I. H. (2020). A renewed look at Greenstone: Entering the third decade. *IPSI Transactions on Internet Research*, *16*(1), 1.

Benedikt, L., Joshi, C., Nolan, L., Henstra-Hill, R., Shaw, L., & Hook, S. (2020). Human-in-the-loop AI in government: A case study. *Proceedings of the 25th International Conference on Intelligent User Interfaces*, 488–497.

Bredin, H., & Laurent, A. (2021). End-to-end speaker segmentation for overlap-aware resegmentation. *Proc. Interspeech 2021*.

Bredin, H., Yin, R., Coria, J. M., Gelly, G., Korshunov, P., Lavechin, M., Fustes, D., Titeux, H., Bouaziz, W., & Gill, M.-P. (2020). Pyannote. audio: Neural building blocks for speaker diarization. *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 7124–7128.

Budd, S., Robinson, E. C., & Kainz, B. (2021). A survey on active learning and human-in-the-loop deep learning for medical image analysis. *Medical Image Analysis*, *71*, 102062.

Carrillo, H., & Lipman, D. (1988). The multiple sequence alignment problem in biology. *SIAM journal on applied mathematics*, *48*(5), 1073–1082.

Chen, Y.-C., & Chen, C.-P. (2009). A framework for machine translation output combination. *ROCLING 2009 Poster Papers*, 309–318.

Chiron, G., Doucet, A., Coustaty, M., Visani, M., & Moreux, J.-P. (2017). Impact of OCR errors on the use of digital libraries: Towards a better access to information. *2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, 1.

Christen, K. (2015). Tribal archives, traditional knowledge, and local contexts: Why the "s" matters. *Journal of western archives.*, *6*(1).

Christen, K. A. (2012). Mukurtu: An Indigenous archive and publishing tool.

Chrons, O., & Sundell, S. (2011). Digitalkoot: Making old archives accessible using crowdsourcing. *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence*.

Cohen, D. (2008). Introducing Omeka.

Corlosquet, S., Delbru, R., Clark, T., Polleres, A., & Decker, S. (2009). Produce and consume linked data with Drupal! *International Semantic Web Conference*, 763–778.

Cundiff, M. V. (2004). An introduction to the metadata encoding and transmission standard (METS). *Library Hi Tech*, *22*(1), 52–64.

Daniel, L., Ferguson, J., Gray, T., Harvey, A., Harvey, D., Pachtner, D., & Troost, K. (2011). Engaging with library users: Sharpening our vision as subject librarians for the Duke University Libraries. *Collections and User Services Council, January*, *14*.

Daniels, C., Holtze, T. L., Howard, R. I., & Kuehn, R. (2014). Community as resource: Crowdsourcing transcription of an historic newspaper. *Journal of electronic resources librarianship*, *26*(1), 36–48.

Dey, R., Balabantaray, R. C., Mohanty, S., Singh, D., Karuppiah, M., & Samanta, D. (2022). Approach for preprocessing in offline optical character recognition (OCR). *2022 Interdisciplinary Research in Technology and Management (IRTM)*, 1–6.

Došilović, F. K., Brčić, M., & Hlupić, N. (2018). Explainable artificial intelligence: A survey. *2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO)*, 0210–0215.

Durniak, A. (2000). Welcome to IEEE Xplore. *IEEE Power Engineering Review*, *20*(11), 12.

Edgar, R. C., & Batzoglou, S. (2006). Multiple sequence alignment. *Current opinion in structural biology*, *16*(3), 368–373.

Estellés-Arolas, E., & González-Ladrón-de-Guevara, F. (2012). Towards an integrated crowdsourcing definition. *Journal of Information science*, *38*(2), 189–200.

Fiscus, J. G. (1997). A post-processing system to yield reduced word error rates: Recognizer Output Voting Error Reduction (ROVER). *1997 IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings*, 347–354.

Grainger, T., & Potter, T. (2014). *Solr in action*. Manning Publications Co.

Granell, E., & Martínez-Hinarejos, C.-D. (2015). Multimodal output combination for transcribing historical handwritten documents. *International conference on computer analysis of images and patterns*, 246–260.

Hagon, P. (2013). Trove: Crowdsourcing behaviour. *National Library of Australia*. https://www.nla.gov.au/sites/default/files/trove-crowdsourcing-behaviour.pdf

Hain, T., Burget, L., Dines, J., Garau, G., Karafiat, M., Lincoln, M., Vepa, J., & Wan, V. (2006). The AMI meeting transcription system: Progress and performance. *Machine Learning for Multimodal Interaction: Third International Workshop, MLMI 2006, Bethesda, MD, USA, May 1-4, 2006, Revised Selected Papers 3*, 419–431.

Harwood, T. G., & Garry, T. (2003). An overview of content analysis. *The marketing review*, *3*(4), 479–498.

Hilbert, M., & López, P. (2011). The world's technological capacity to store, communicate, and compute information. *science*, *332*(6025), 60–65.

Holley, R. (2009). How good can it get? analysing and improving OCR accuracy in large scale historic newspaper digitisation programs. *D-Lib Magazine*, *15*(3/4).

Holley, R. (2010). Trove: Innovation in access to information in Australia. *Ariadne*, (64).

Hook, D. W., Porter, S. J., & Herzog, C. (2018). Dimensions: Building context for search and evaluation [https://www.frontiersin.org/articles/10.3389/frma.2018.00023 *Frontiers in Research Metrics and Analytics*, *3*, 23. https://doi.org/10.3389/frma.2018.00023

Jakubec, K., Polák, M., Nečaskỳ, M., & Holubová, I. (2014). Undo/redo operations in complex environments. *Procedia Computer Science*, *32*, 561–570.

Kamlah, J., & Stegmüller, J. (2018, November). Ocromore - combining multiple OCR-engine results to improve character recognition accuracy. https://doi.org/10.5281/zenodo.1493860

Kuchaiev, O., Li, J., Nguyen, H., Hrinchuk, O., Leary, R., Ginsburg, B., Kriman, S., Beliaev, S., Lavrukhin, V., Cook, J., et al. (2019). Nemo: A toolkit for building AI applications using neural modules. *arXiv preprint arXiv:1909.09577*.

Lagoze, C., Payette, S., Shin, E., & Wilper, C. (2006). Fedora: An architecture for complex objects and their relationships. *International Journal on Digital Libraries*, *6*, 124–138.

Leggott, M. A. (2009). Islandora: A Drupal/Fedora repository system.

Makridakis, S. (2017). The forthcoming Artificial Intelligence (AI) revolution: Its impact on society and firms. *Futures*, *90*, 46–60.

Martínez-González, M. (2005). Document versioning in digital libraries. In *Encyclopedia of database technologies and applications* (pp. 201–205). IGI Global.

McCrocklin, S., Humaidan, A., & e Edalatishams, I. (2019). ASR dictation program accuracy: Have current programs improved? *Pronunciation in Second Language Learning and Teaching Proceedings*, *10*(1).

McKnight, S. W., Hogg, A. O., & Naylor, P. A. (2021). Analysis of phonetic dependence of segmentation errors in speaker diarization. *2020 28th European Signal Processing Conference (EUSIPCO)*, 381–385.

MediaWiki. (2023). What is MediaWiki? [Accessed: 27-08-2023]. https://www.mediawiki.org/wiki/MediaWiki

Monarch, R. M. (2021). *Human-in-the-loop machine learning: Active learning and annotation for human-centered AI*. Simon; Schuster.

Montenegro, M. (2019). Subverting the universality of metadata standards: The TK Labels as a tool to promote Indigenous data sovereignty. *Journal of Documentation*, *75*(4), 731–749.

Omena, J. J., Pilipets, E., Gobbo, B., Chao, J., et al. (2021). The potentials of Google Vision API-based networks to study natively digital images. *Diseña*, *19*, 1–25.

Osborn, W., Fox, S., Bainbridge, D., & Witten, I. H. (2011). Automating the maintenance of Greenstone collections. In *Digital libraries-methods and applications*. IntechOpen.

Pavlichenko, N., Stelmakh, I., & Ustalov, D. (2021). Crowdspeech and voxdiy: Benchmark datasets for crowdsourced audio transcription.

Prokopalo, Y., Shamsi, M., Barrault, L., Meignier, S., & Larcher, A. (2021). Active correction for speaker diarization with human in the loop. *Interspeech*.

Raj, D. (2023, April). https://github.com/desh2608/dover-lap

Raj, D., Garcia-Perera, L. P., Huang, Z., Watanabe, S., Povey, D., Stolcke, A., & Khudanpur, S. (2021). Dover-lap: A method for combining overlap-aware diarization outputs. *2021 IEEE Spoken Language Technology Workshop (SLT)*, 881–888.

Ryant, N., Church, K., Cieri, C., Cristia, A., Du, J., Ganapathy, S., & Liberman, M. (2018). First DIHARD challenge evaluation plan. *2018, tech. Rep.*

Salvaris, M., Dean, D., & Tok, W. H. (2018). Deep learning with Azure. *Building and Deploying Artificial Intelligence Solutions on Microsoft AI Platform, Apress.*

Sato, T., Kanade, T., Hughes, E. K., Smith, M. A., & Satoh, S. (1999). Video OCR: Indexing digital news libraries by recognition of superimposed captions. *Multimedia Systems*, *7*, 385–395.

Singh, A., Bacchuwar, K., & Bhasin, A. (2012). A survey of OCR applications. *International Journal of Machine Learning and Computing*, *2*(3), 314.

Sinha, Y., & Siegert, I. (2021). Improving the accuracy for voice-assistant conversations in German by combining different online ASR-API outputs. *Human Perspectives on Spoken Human-Machine Interaction*, 11–16.

Staples, T., Wayland, R., & Payette, S. (2003). The Fedora Project. *D-Lib Magazine*, *9*(4), 1082–9873.

Stolcke, A., & Yoshioka, T. (2019). DOVER: A method for combining diarization outputs. *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 757–763.

Swanson, A., Kosmala, M., Lintott, C., Simpson, R., Smith, A., & Packer, C. (2015). Snapshot Serengeti, high-frequency annotated camera trap images of 40 mammalian species in an African savanna. *Scientific data*, *2*(1), 1–14.

Tanner, S. (2001). Librarians in the digital age: Planning digitisation projects. *Program*, *35*(4), 327–337.

Tesseract-OCR. (2023, May). https://github.com/tesseract-ocr/tessdoc/blob/main/ImproveQuality.md

Veridian user text correction. (2023). https://veridiansoftware.com/knowledge-base/user-text-correction-guidelines/

Walker, J., Fujii, Y., & Popat, A. C. (2018). A web-based OCR service for documents. *Proceedings of the 13th IAPR international workshop on document analysis systems (DAS), Vienna, Austria*, *1*.

Weibel, S., Kunze, J., Lagoze, C., & Wolf, M. (1998). *Dublin Core metadata for resource discovery* (tech. rep.).

Wemhoener, D., Yalniz, I. Z., & Manmatha, R. (2013). Creating an improved version using noisy OCR from multiple editions. *2013 12th International Conference on Document Analysis and Recognition*, 160–164.

White, J. (2020). Pubmed 2.0. *Medical reference services quarterly*, *39*(4), 382–387.

Witten, I. H., Bainbridge, D., & Nichols, D. M. (2009). *How to build a digital library*. Morgan Kaufmann.

Witten, I. H., Bainbridge, D., Paynter, G., & Boddie, S. (2002). The Greenstone plugin architecture. *Proceedings of the 2nd ACM/IEEE-CS joint conference on digital libraries*, 285–286.

Witten, I. H., Moffat, A., & Bell, T. C. (1999). *Managing gigabytes: Compressing and indexing documents and images*. Morgan Kaufmann.

Wooters, C., & Huijbregts, M. (2007). The ICSI RT07s speaker diarization system. *International Evaluation Workshop on Rich Transcription*, 509–519.

Wu, J., Williams, K., Khabsa, M., & Giles, C. L. (2014). The impact of user corrections on a crawl-based digital library: A CiteSeerX perspective. *10th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*, 1–5.

Wu, J., Williams, K. M., Chen, H.-H., Khabsa, M., Caragea, C., Tuarob, S., Ororbia, A. G., Jordan, D., Mitra, P., & Giles, C. L. (2015). Citeseerx: AI in a digital library search engine. *AI Magazine*, *36*(3), 35–48.

Yalniz, I. Z., & Manmatha, R. (2011). A fast alignment scheme for automatic OCR evaluation of books. *2011 International Conference on Document Analysis and Recognition*, 754–758.