Segmenting Electronic Theses and Dissertations By Chapters

Javaid Akbar Manzoor

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Science and Application

Edward A. Fox, Chair
Lenwood S. Heath
Jian Wu

September 23, 2022
Blacksburg, Virginia

Keywords: segmentation, deep-learning, natural language processing, ETD, digital libraries

Segmenting Electronic Theses and Dissertations By Chapters

Javaid Akbar Manzoor

ABSTRACT

The ability to determine the underlying document structure of Electronic Theses and Dissertations (ETDs) can facilitate the development of many services. Some of these include summarizing long documents, semantic parsing, classification, and indexing for discovery. This thesis puts forward a bottom-up approach to segmenting ETDs and shows that LaTeX manipulation can be used to help create meaningful data sets. The research provides two data sets. The first contains 1,459 ETDs plus the chapter boundaries generated using our Chapter-Segmentation Pipeline. The second is a data set containing 150 ETDs with manually labelled chapter boundaries. This research also involves training deep learning models using our data sets. Overall, the goals of this research are to encourage researchers within the natural language processing community to incorporate the power of LaTeX manipulation to create more data sets, and to increase ETD accessibility.

Segmenting Electronic Theses and Dissertations By Chapters

Javaid Akbar Manzoor

GENERAL AUDIENCE ABSTRACT

Electronic theses and dissertations (ETDs) are structured documents in which chapters are major components. There is a lack of any repository that contains chapter boundary details alongside these structured documents. Revealing these details of the documents can help increase accessibility. This research explores the manipulation of ETDs marked up using LaTeX to generate chapter boundaries. We use this to create a data set of 1,459 ETDs and their chapter boundaries. Additionally, for the task of automatic segmentation of unseen documents, we prototype three deep learning models that are trained using this data set. We hope to encourage researchers to incorporate LaTeX manipulation techniques to create similar data sets.

Dedicated to Baap and Ma, Appi and Ali, Rajuh and Nabeel.

Acknowledgments

First and foremost, I need to express my gratitude to my parents for supporting me throughout my time as a student. I cannot thank Professor Edward A. Fox enough for the advice and the resources that he brought to my attention; they have been crucial to my success as a researcher. I would also like to mention the incredibly resourceful committee members, Professor Lenwood Heath and Professor Jian Wu. Needless to say the resources and advice they provided were vital in creating my segmentation data sets. Lastly, it would be remiss of me to not mention the IMLS project related team at Virginia Tech, and the support of IMLS through grant IMLS LG-37-19-0078-19. The team's unique perspective and ability to foster an encouraging environment helped me greatly as a researcher interested in the world of Electronic Theses and Dissertations.

Contents

List of Figures

List of Tables

List of Abbreviations

ETDs Electronic Theses and Dissertations

NLP Natural Language Processing

CSE Chapter-Segmented ETD

CSP Chapter-Segmentation Pipeline

PNG Portable Network Graphics

POS Part-of-Speech

Chapter 1

Introduction

1.1 Background

Electronic Theses and Dissertations (ETDs) contain a vast amount of knowledge that could be focused on a single discipline or could be interdisciplinary. Each document can contain diverse knowledge based on its discipline(s). This can also be the case with each chapter [15]. A chapter discussing the data being used can have high variance in vocabulary compared to a chapter discussing how this data is being processed. This specificity suggests that allowing search at the chapter level might be helpful to scholars. Revealing the underlying document structure through identifying each chapter can help users find meaningful content within theses, and increase accessibility at higher granularity. This thesis focuses on segmenting these documents into chapters in an effort to increase accessibility.

Using machine learning and natural language processing, researchers have created complex architectures that address modern research questions such as semantic processing, automatic summarization, text classification, mining relevant information, and much more. The related models have shown state-of-the-art results for their tasks when applied to short documents, but they struggle with long documents [29]. These challenges come as a consequence of limitations of deep learning with long sequence lengths [42]. One of those limitations is the attention mechanism within deep learning models having difficulty retaining long sequence dependencies [27].

There has been a lack of data sets containing documents with their chapter boundary details. Available ETD repositories usually only contain limited metadata information such as title, author, institution, degree, year, and abstract – along with a PDF version of the document.

ETDs divided into chapters can assist the research community. Access to more granular content within these documents can help librarians, researchers, and students make more specific queries. There are also many ETDs that are interdisciplinary in nature, where the knowledge may vary widely between chapters [15].

This research explores segmenting scholarly documents such as theses and dissertations into chapters. Access is provided to multiple data sets. This includes a repository containing 1,459 ETDs with chapter boundaries generated using our Chapter-Segmentation Pipeline (CSP), and another data set containing 150 ETDs with manually labelled chapter boundaries. The latter can be used as ground truth data. These data sets were created using the help of the open-access arXiv repository. The arXiv repository provides documents as PDFs along with the LaTeX files that were used to create those PDFs. LaTeX files were leveraged to help collect accurate structural information of the chapter boundaries of each document. The different ways LaTeX was manipulated in order to output chapter boundary details and the challenges that were faced, are also discussed. This work provides a foundation for researchers to help improve upon implemented LaTeX manipulation techniques, machine learning algorithms for longer documents, and the integration of deep learning models to segment new scholarly documents for more granular accessibility.

1.2 Motivation

Scholarly documents such as theses and dissertations contain valuable information organized into chapters. All chapters are related to the overall subject matter, but each chapter may

employ different vocabulary depending on the subtopic being discussed. The goal is to break down theses and dissertations into chapters to reveal the underlying structure of these documents. This can help increase accessibility to the information within these documents and reveal interrelationships of chapters within theses.

Access to chapter-level information for ETDs can help assist users in many ways. Researchers, students, and teachers can quickly search documents and specifically view chapters to answer specialized queries. This research can increase access to interdisciplinary ETDs that could be helpful for browsing as well. Furthermore, the Chapter-Segmentation Pipeline (CSP) can contribute to the research within the deep learning and natural language processing (NLP) communities. Many of the current state-of-the-art natural language models work well on shorter documents but show poor results on longer documents. Tasks such as automatic summarization, language modelling, and part-of-speech (POS) tagging rely on sequences using an attention mechanism. The attention mechanism works poorly on longer sequences as a result of vanishing gradients. Dividing longer documents into more specific chapters can help create meaningful batches to train suitable models.

1.3 Problem Statement and Solution

The problem is how to segment a born-digital PDF file for an ETD into chapters using an automated approach.

The first goal is to create and collect chapter boundary details for Electronic Theses and Dissertations (ETDs) by manipulating source LaTeX code. To accomplish this task, a repository containing ETDs that comes with source LaTeX files for each document was used. The LaTeX code for each thesis must accurately regenerate the original PDF so chapter-boundary information is 1-to-1 to what has already been published. Given an ETD, the Chapter-

Segmentation Pipeline generates chapter-boundary details with page-ranges for each chapter. arXiv's repository [39] was used to help create this data set. An altered PDF version is generated by modifying provided LaTeX files. This altered PDF will allow us to generate the chapter boundary details.

The second goal is to use the results from the first goal, applied to the 1,459 ETD corpus from arXiv, to train a deep learning model that then can be applied to other PDF files that are born-digital ETDs. Such a high quality model then can be used to extract chapters from a large ETD corpus, allowing further research on chapter summarization, chapter classification, chapter topic analysis, and searching at the chapter level.

1.4 Research Questions

To address our research problem, we sought to answer the following research questions.

1. RQ1: Can LaTeX manipulation of ETDs, that are marked up using LaTeX, be relied upon to create new meaningful data sets that aid identification of chapter boundaries?

    - LaTeX is being used frequently to write scholarly documents. Can manipulating LaTeX representations of ETDs allow us to collect important details within these documents reliably?

2. RQ2: How well do existing methods perform in segmenting the ETD documents into chapters?

    - There exist many tools that output chapter boundary information, but the accuracy of these tools has yet to be explored for ETDs.

3. RQ3: Can we create a reliable Gold Standard data set for long documents (i.e., ETDs) segmented by chapters?

    - Since there is a lack of any data set that describes the correct chapter segmentation of long documents, i.e., adds their structural information as well, it would be incredibly useful to create one. It would be interesting to see if, using the power of LaTeX, accurate information can be provided.

4. RQ4: Can sequence labelling techniques using deep learning be applied to documents by labelling each page according to its chapter sequence?

    - There has been a great deal of research on part-of-speech tagging within the natural language processing community. Words in a sentence are treated as a sequence, and each word is labelled with a POS tag. Models incorporate the ordering information to tag each word with an appropriate sequence tag. In a similar fashion, can deep learning models learn to label each page of a scholarly document using chapter labels?

5. RQ5: Can the performance of a model trained on this Chapter-Segmented ETD (CSE) data set perform well on new born-digital ETD PDF documents?

    - The efficiency of a model that can segment long documents is only useful if it can represent the vast variety of scholarly documents in the research space. Can a segmentation model perform well on theses and dissertations it has not seen before?

1.5 Research Challenges

1.5.1 Lack of Gold Standard Data

There are no publicly available repositories that contain ETDs along with details of their underlying document structures. Training machine learning models to automatically segment long documents into chapters would require suitable labeled data. Since there is no data set available, our challenge was to create such a data set, which we conjectured could be done using the open-access distribution of arXiv and its provided source LaTeX files.

1.5.2 LaTeX Modifications

arXiv provides open access to over 1.8 million scholarly documents. There have been prior efforts to manipulate LaTeX to create meaningful training data from scholarly documents [16, 26]. These studies have used the arXiv and PubMed data sets to create high-quality labels for the task of figure extraction. They use the additional information these repositories provide within auxiliary files to locate figures. They modified the source LaTeX files to output bounding boxes around the figures. Modifying LaTeX files can be tricky because of the large number of classes and versions present within LaTeX's typesetting environment. We conjecture that suitable manipulation of ETDs made available through arXiv could lead to the training data we sought.

1.5.3 Training Deep Learning Models

Given that there are not a large number of ETDs available in arXiv and given that deep learning models must be trained with large amounts of example data, it was unclear if our

approach would yield enough training data to build good models that could be applied to other born-digital ETD PDF files. We conjectured that a suitable deep learning architecture could be found that would lead to high quality models with good predictive performance on new ETD PDF files.

1.6 Research Contributions

Our work makes the following contributions:

- Provides a new data set of 1,459 ETDs with chapter boundaries labelled using our Chapter-Segmentation Pipeline.
- Provides a new manually labelled data set of 150 ETDs with chapter boundary labels.
- Describes modification of LaTeX files to output details while it generates PDF files at compile time.
- Provides deep learning models that can help automatically segment new documents into chapters.

1.7 Thesis Outline

This thesis is organized as follows:

Chapter 2 discusses background information regarding related works and prior knowledge needed to understand this research.

Chapter 3 discusses the background, characteristics, and origins of the data we have used. It follows this by discussing the preparation of the original ETD PDF document using the source files.

Chapter 4 discusses the Chapter-Segmentation Pipeline in detail. It describes the steps taken to generate accurate chapter boundary details.

Chapter 5 discusses the data set generated by the Chapter-Segmentation Pipeline. It discusses manual inspection performed to validate the CSP's results. It follows this by discussing another manually labelled data set that was used to evaluate deep learning models.

Chapter 6 discusses the different types of deep learning models that were designed. It discusses validation splits, expected inputs and outputs for each model, and the design of experiments.

Chapter 7 discusses the results of the various experiments that were performed on the deep learning models. It also compares the LaTeX manipulation pipeline to other segmentation tools.

Finally, Chapters 8 and 9 conclude the thesis with limitations, conclusions that can be drawn from experimentation, and suggestions on possible future directions.

Chapter 2

Literature Review

This chapter focuses on discussing relevant tools and techniques that have been used in this research. The literature surveyed provides a background for the content in the following chapters.

2.1 Deep Learning

Deep learning is a kind of machine learning that employs neural networks. They contain input layers, one or more hidden layers, and output layers. Each layer contains nodes that are connected to the nodes in the following layers. Nodes and their connections contain weights and biases [25] [34]. Upon training a deep learning model, errors from forward layers are back-propagated towards previous layers to reduce or minimize a cost function [21]. This allows models to continually train and improve over iterations. Figure 2.1 displays a very simple neural network with 3 nodes in the input layer, 2 hidden layers (each with 4 nodes), and one output layer.

2.1.1 Image Features

Deep learning architectures such as the one displayed in Figure 2.1 are not ideal for tasks with inputs whose vector representations reflect high dimensionality. Text, images, and

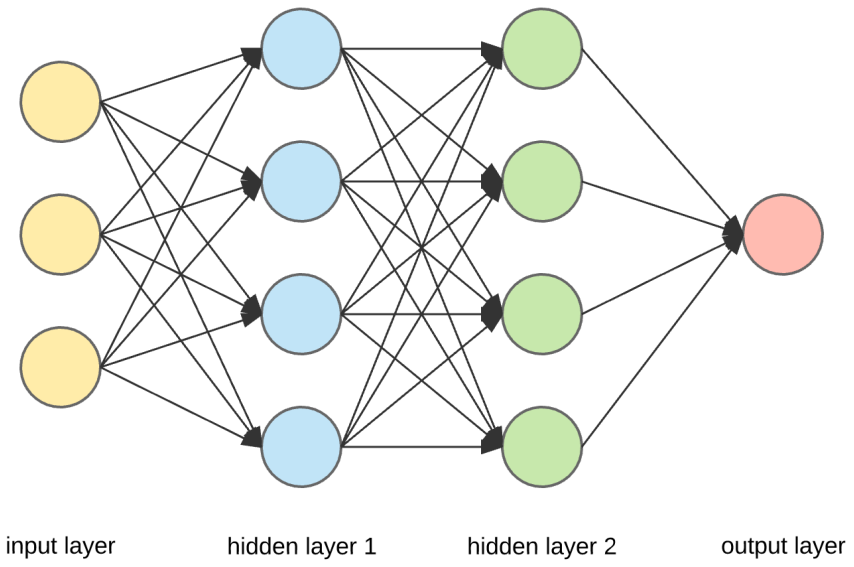


Figure 2.1: Simple neural network architecture adapted from [11]

videos must be converted into machine-readable formats leading to smaller vectors known as embeddings [43]. Embeddings allow representing high dimensional data in a lower dimensional space. Dimensionality can be further reduced using convolutional neural networks (CNNs). CNNs have been used to convert and extract high-level features from input data (see additional details in Chapter 6) [24]. Thus, input dimensions can be significantly reduced, leading to representations that retain important information from the original data. Figure 2.2 displays a CNN architecture extracting features from a given image. Extracted image features can be used as inputs in other deep learning models [1, 18].

AlexNet

AlexNet [19] was first published in 2012 and became a standard for visual recognition. It is a leading deep learning architecture containing 8 layers that can be used for image classification tasks. It has pre-trained fully-connected layers or FC7 layers that contain rich feature representations that can be used for object detection tasks. While it was a

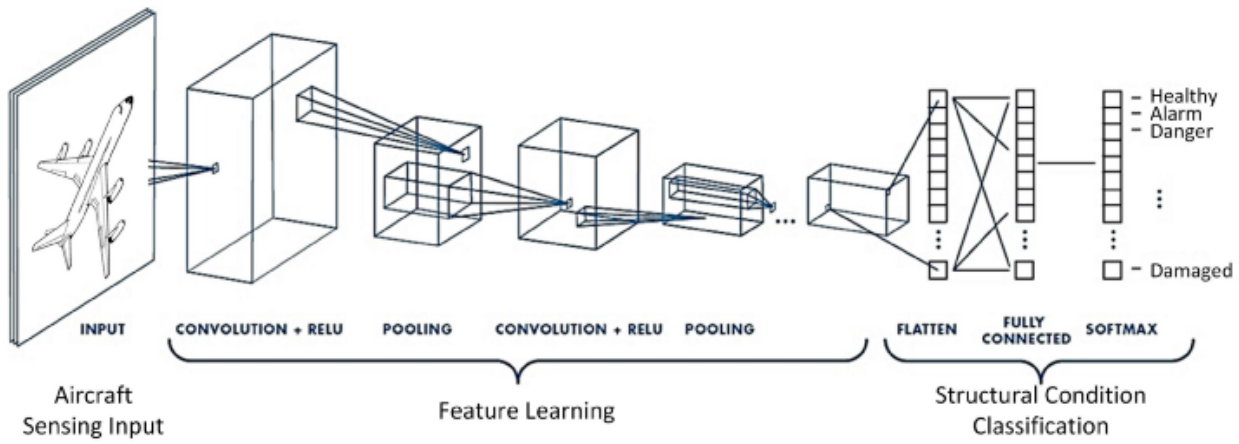


Figure 2.2: Convolutional neural network adapted from [36]

breakthrough back in 2012, there have been variations that build upon the success of AlexNet for richer feature representations [1].

VGG

Visual Geometry Group or VGG is based on AlexNet but further builds upon it. Its name contains the number of layers; for example VGG-16 is 16 layers deep while VGG-19 has 19 layers [35]. Its performance in object recognition models has been ground breaking. Through transfer learning, pre-trained VGG models can be applied in analysis of new collections of images.

2.1.2 Text Embeddings

Text must be converted into vectors to be input to a machine learning model. Text is usually converted into embeddings that are dense vectors with a chosen **N** dimensions. These vectors contain important contextual information for each word and also capture positional relationships to other words.

GloVe

GloVe [31] was developed by Stanford for generating word embeddings using an unsupervised learning model. GloVe embeddings are based upon leveraging a global word-to-word count. Pre-trained GloVe embeddings can be used and further trained for a specific task.

FastText

Facebook developed FastText for efficient word representation learning. It supports supervised classification tasks or unsupervised training [7]. FastText embeddings learn word representations using **n**-grams.

2.2 Tools and Services

This section discusses additional relevant tools and services.

2.2.1 File Formats

DVI

The device independent (DVI) file format is the original output file format of the TeX typesetting program [20]. DVI files are not intended to be human-readable and are usually converted further into something readable. pdfLaTeX [37] is a special variation of the original TeX typesetting environment that directly converts the output into a PDF file.

PDF

Portable Document Format (PDF) was first released in 1992 by Adobe. It is a file format that can be used to represent documents and images and has provided users with an easy way to create, share, and view documents. PDF is now standardized by the International Organization for Standardization (ISO) and its use is widespread. It can contain links and buttons now as well [12].

PNG

Portable Network Graphics (PNG) is a widely used standardized file format used for images [33].

2.2.2 Text Extraction

This subsection reviews various Python [41] packages that can be used to efficiently extract text from PDF documents.

PyMuPDF

PyMuPDF was developed by Artifex Software [4]. It was designed to be an E-Book viewer, renderer, and a toolkit for PDFs and similar formats. It can be used to load PDF documents and extract useful information. This includes page counts and page text. It is a light-weight package and is easy to set up. It also provides users with available metadata information on loaded PDF files.

PyPDF2

PyPDF2 is a free-to-use, open-source Python package. It provides users with the ability to merge, split, and transform PDF documents. It also collects metadata information on loaded PDF files and allows text extraction [13].

Tika-Python

To use Tika-Python, you must have Java 7+ installed on your computer. It is a port that allows using Apache Tika services through a REST server. This toolkit allows users to extract any available metadata information from various file extensions such as PDF and PPT. It also allows users to extract text [23].

2.3 Typesetting

This section discusses various typesetting related tools and services.

2.3.1 LaTeX

LaTeX is used to typeset documents from marked up text. It has been used primarily to write scholarly documents, but also other types of documents [20]. Authors use a markup language to write the document, which is then generated using a TeX processor.

Common TeX distribution services include TeX Live [32], MikTeX, or even online services such as Overleaf [14]. LaTeX comes with various templates, packages, and styles. Users are not only able to write or insert images, but also draw equations, images, and graphs using

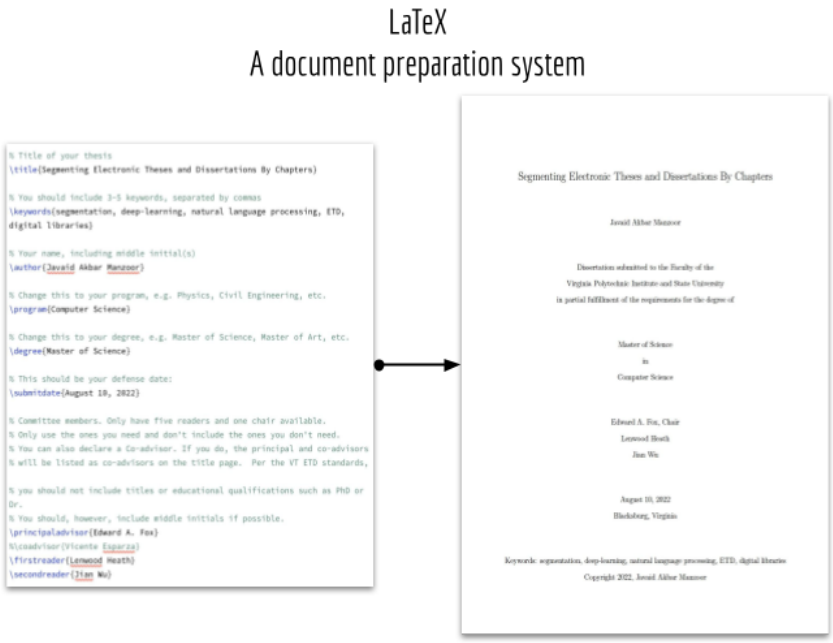


Figure 2.3: LaTeX: Converting from marked up to rendered document.

LaTeX. Over the course of its lifetime, there have also been LaTeX variations developed, which will be discussed next.

2.3.2 pdfLaTeX

The original TeX typesetting environment compiles and outputs a document into a custom format known as DeVice Independent (DVI) format. Later, PDF formats became standard as they allow additional features such as hyperlinks between sections. pdfLaTeX is a similar typesetting environment, but its compiler directly outputs PDF files [37].

2.3.3 XeLaTeX

XeLaTeX, similar to pdfLaTeX, is another TeX typesetting environment that was made to increase compiler functionality. With its introduction, users had support of a wider range of fonts, formats, and range of characters beyond ASCII. It can be used to write documents in numerous languages [17].

2.4 Chapter Segmentation

This section discusses existing tools that provide segmentation capabilities. Section 7.1 discusses the results of segmentation using these tools.

2.4.1 GROBID

GROBID uses machine learning to extract, parse, and process PDF documents. It generates output as XML/TEI [9] encoded documents and has been in widespread use mainly for scholarly documents. Final XML/TEI files include metadata information, segmentation details, and citation details [22].

2.4.2 Intelligent Core Systems

Intelligent Core Systems, or ITCORE, can process PDF files and generate structural information such as chapters, subsections, and other textual information. It uses heuristic rules that utilize the table of contents to generate details [2, 3]. It requires Java 7+ installed and can be accessed either as a package or using a web application. The final outputs of ITCORE are TEI encoded in a hierarchical structure.

Chapter 3

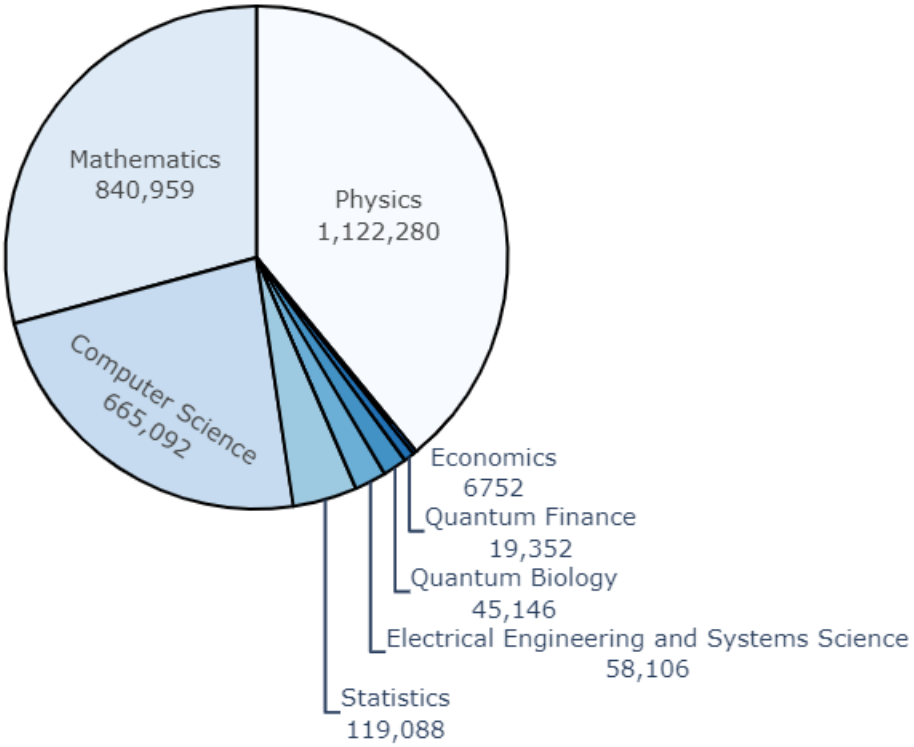
LaTeX Source Files

This chapter begins with an overview of the source data. It follows that with a description of LaTeX documents and the importance of detecting each document's main TeX file. Lastly, it discusses generating copies of original PDFs by compiling LaTeX source files.

3.1 arXiv.org

arXiv is an open-access archive for scholarly articles. It contains over **1.8 million** scholarly articles from a broad range of disciplines [39]. These documents include information on topics relating to physics, mathematics, computer science, quantitative biology, quantitative finance, statistics, electrical engineering, systems science, and economics. The pie chart in Figure 3.1 shows the number of documents in each primary category on arXiv. This data was collected from arXiv's free to use metadata available on *Kaggle* by Cornell University [40].

Each main category has its own set of subcategories to increase accessibility for specialized queries. The documents take the form of short articles, reports, and even long book-styled dissertations.



Category	Documents	Percentage
Physics	1,122,280	39.0
Mathematics	840,959	29.2
Computer Science	665,092	23.1
Statistics	119,088	4.14
Electrical Engineering and Systems Science	58,106	2.02
Quantum Biology	45,146	1.57
Quantum Finance	19,352	0.673
Economics	6752	0.235

Figure 3.1: arXiv: Document distribution for primary categories [40]

3.1.1 Source Files and Directory Structure

arXiv provides bulk access to uploaded documents through Amazon S3. Each document comes with its source files and the resulting PDF file. For the creation of our Chapter-Segmented ETD data set these source files were vital as they include the LaTeX files that were used to generate the PDF. Previously, Virginia Tech Ph.D. candidate Bipasha Banarjee extracted and manually verified **2,838** ETDs from arXiv. This had to be done manually because arXiv does not include any separate definitive metadata details regarding document type. A document could be a report, article, or thesis, for example. Since our primary interest is creating a segmentation repository for ETDs only, her work came as a blessing. Thus, this extracted repository was used to create the Chapter-Segmented ETD data set. The extracted documents come in an organised directory structure labelling each document's folder using a key in the following format: **YYMM.UniqueID**. On arXiv these unique IDs can be accessed by including a collection prefix. The list of prefixes include:

- astro-ph
- cond-mat
- gr-qc
- hep-ex
- hep-lat
- hep-ph
- hep-th
- math-ph
- nlin
- nucl-ex
- nucl-th
- physics
- quant-ph
- math
- CoRR

These prefixes were later added to the metadata for the records. Each folder contains LaTeX source files, figures, and processed PDFs. Each folder can have a different structure. Figure 3.2 shows the directory structure for a few sample ETDs. Files highlighted in red are the processed PDFs of each thesis or dissertation. These PDFs are often not present within the

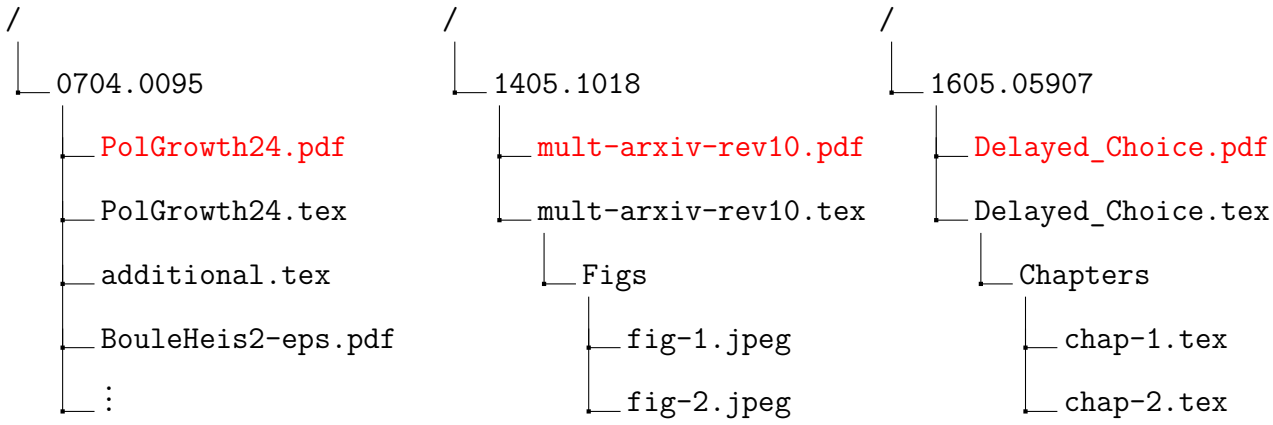


Figure 3.2: Directory structures in arXiv bulk-access

source file and must be generated by compiling the main TeX file. They were generated using TeX files with the same file name. We will discuss our detection process for the main TeX file in Section 3.3.1.

3.2 LaTeX Document Structure

Before we discuss LaTeX modification, it would help to discuss the general structure of a LaTeX file used to generate processed PDFs. Figure 3.3 shows a code snippet for a simple LaTeX document.

Each document begins with a document type declaration as shown in line 1 of Figure 3.3. LaTeX provides users with a number of classes to pick from, ranging from articles to books to theses. Classes and packages allow users to create documents in various different styles. Additional parameters such as font-size can be passed to this command to effect the overall document. Section 3.2.1 discusses how this command helped in finding the main TeX file of each ETD.

```
1  \documentclass[11pt]{article}
2
3  % Preamble Section
4  % Here you define the type of document you are writing, the
5  % language you are writing in,
6  % the packages you would like to import, and much more.
7
8  \begin{document}
9
10 % Here you write the actual content of the document. It can
11 % include LaTeX code to declare
12 % environments such as figures, tables, and much more.
13
14 % If you have organised your document using multiple TeX files,
15 % this is also where you could
16 % import each file using the \include{filename.tex} command.
17
18 \end{document}
```

Figure 3.3: Sample LaTeX code snippet

The commands between line 1 and line 7 in Figure 3.3 are said to constitute the preamble section of a document. This section allows authors to import packages that can be used for various purposes like styling or positioning of key elements such as figures, tables, or equations. Users can alter the behaviour of native LaTeX commands in this section. This is also where code was inserted to manipulate behaviour during compile time to provide chapter boundary details. This will be discussed in detail in Chapter 4.

Finally, lines between line 7 and line 15 are where users would add the contents of the document. Users can create a variety of environments, declare sections, and much more. If users have created multiple TeX files for different sections, this is also where they would be imported.

Listing 3.1: Main.tex

```
1  \documentclass[11pt]{
       article}
2  .
3  .
4  .
5  \begin{document}
6  .
7  .
8  \include{Chap_1.tex}
9  \include{Chap_2.tex}
10 .
11 .
12 \end{document}
```

Listing 3.2: Chap_1.tex

```
1 \chapter{Introduction}
2 .
3 .
4 Contents of Chapter 1
5 .
6 .
```

Listing 3.3: Chap_2.tex

```
1 \chapter{Chapter 2}
2 .
3 .
4 Contents of Chapter 2
5 .
6 .
```

Figure 3.4: Main LaTeX file including other TeX files

3.2.1 Main TeX File Detection

Authors using LaTeX can write the entire document in a single TeX file. However, working with a single TeX file can make the writing process confusing. Typically, authors divide the contents of their writing over multiple TeX files. The overall document is brought together using a central or main TeX file. This file must contain the declarations of the document environment as shown on line 1 in Figure 3.3. Declaration of the document environment is imperative as the topmost environment within LaTeX.

For any kind of modification to be possible, detecting the main TeX file is crucial. It also allows one to compile the original document. Each document's source directory could have varied structures depending on the original author's discretion. Figure 3.2 shows sample structures for 3 documents. Each document has a different organization style with additional files and folders. When the main TeX file is compiled, it brings all these files together to form a new processed PDF. The structure of these directories must be kept intact for any modification to work. The main TeX file can then include other TeX files using commands within the content section. Figure 3.4 shows a simple demonstration of this.

While using the **documentclass** command to detect main TeX files for each document seems like a straightforward solution there were a few complications that had to be addressed which are described below.

Exceptions

DocumentStyle Command

In previous versions of LaTeX users could use a **documentstyle** command to declare the environment. This is however an outdated command, and environments created using this could not run our injected code. Since this command operates in the same manner as **documentclass**, fixing this meant simply replacing the text **documentstyle** with **documentclass**.

Comments

In LaTeX, comments can be made using the % character if inserted at the start of a line. Any text in the line after this command will be ignored by the compiler. Many of the collected documents had comments that referenced the **documentclass** command. To avoid any issues, all comments were removed using a Python script, for all TeX files.

Subfile and Standalone

LaTeX contains special parameters that could be passed to the **documentclass** command that could alter the behaviour of the TeX file.

```
\documentcIass{subfile}
\documentcIass{standalone}
```

```
/** The "I" is a stand-in for "l" to avoid LaTeX compile issues. **/
```

The **subfile** command can also be used to create multi-file LaTeX projects. Once users have a main TeX file created, all sub-files can be linked to the main TeX file using this command. The **standalone** parameter can be passed to create a standalone TeX file. This is primarily used for TeX images and TeX code which can then be included within the main file. These commands were ignored while detecting potential candidates for main TeX files by using regular expressions.

3.3 Generating Original PDF

Generating the original PDF is crucial for the Chapter-Segmentation Pipeline discussed in Chapter 4. As aforementioned, thanks go to fellow researcher and Ph.D. candidate Bipasha Banarjee, for she had manually collected and verified the source files for 3118 ETDs from arXiv. However, 280 duplicates were detected among the 3,118, so the usable set of source files resulting numbered 2,838. This set of documents formed the basis of our research. However, the original PDF had to be generated by compiling the source files first. LaTeX files were preprocessed by removing all comments using **Python** [41]. This was done to avoid issues when scanning for main TeX files.

3.3.1 Detecting Main TeX File

For a given ETD, its source folder were filtered for TeX files. These TeX files were scanned for the main TeX file using the regular expressions displayed below.

```
(?<!\%)(?<!\%\)\\\\documentstyIe
(?<!\%)(?<!\%\)\\\\ documentcIass
```

```
/** The "I" is a stand-in for "l" to avoid LaTeX compile issues. **/
```

Additionally any files declared as a subfile or standalone document were ignored using the following regular expressions.

```
(?<!%)\\\\documentcIass\s?\{subfile\}
(?<!%)\\\\documentcIass\s?\{standalone\}
```

Table 3.1 shows metrics relating to the detection process. Any LaTeX files that used the **documentstyle** command were changed to use the **documentclass** command instead.

	Number of ETDs
Total ETDs	2,838
Detected Successfully	2,610
Failed Detection	228
Used DocumentStyle Command	131
Subfile Files Found	62

Table 3.1: Main TeX file detection metrics

As mentioned in Table 3.1, the detection scheme failed to detect the main TeX file for 228 ETDs. These also include ETDs that had conflicting results between multiple TeX files. If the detection scheme qualified more then one TeX file within an ETD source file as a main TeX file, they were labelled "conflicted". Finally, only ETDs with successful detection for a main TeX file were moved forward towards generating the original PDF.

3.3.2 Compiling Documents

This work was conducted on a machine using a Linux operating system. A full install of *TeX-Live* [32] was performed to include all packages. All packages were installed to account

for any packages being used by the source files. Lastly, pdfLaTeX [37] was the TeX engine used to compile the main TeX files. Out of 2,610 source files, **1,872** successfully generated a file identical to the original PDF. If an ETD's source LaTeX files fail to generate the same as the original PDF, it is not sent forward to generate the chapter details. Table 3.2 summarizes the list of errors encountered for source files that failed to generate the original PDF.

Error	Description	Files
no legal \end found	Missing end } or \end of element	151
Enter file name:	Missing an imported file, .sty, or .cls file	137
Over 100 Errors	Too many consecutive errors	403
Error Reading Image File	Faulty or missing image files	3
Please type another input file name	Faulty input file	11
Need more memory	TeX engine ran out memory	17
Others	Abruptly ended during compile time	16

Table 3.2: Issues encountered while compiling original PDF

There were in total **738** source folders that failed to generate the original PDF. There were many source files with faulty code or missing files.

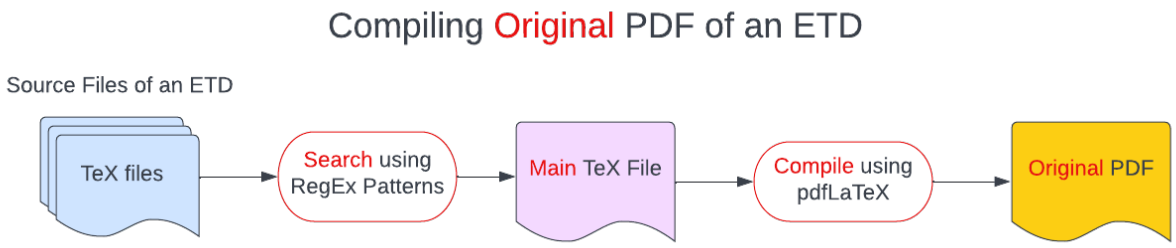


Figure 3.5: Generating Original PDF

Figure 3.5 shows a summary of each step towards creating the original PDF for a single ETD. Eventually, from a set of a 2,838 source files, we were able to obtain **1,872** source files of ETDs with workable LaTeX source files.

Chapter 4

Chapter-Segmentation Pipeline

This chapter begins with an overview (see Figure 4.1) of the Chapter-Segmentation Pipeline (CSP) and motivation for choosing it, after other prior attempts. Then follows a detailed explanation of steps taken to create accurate chapter boundary details.

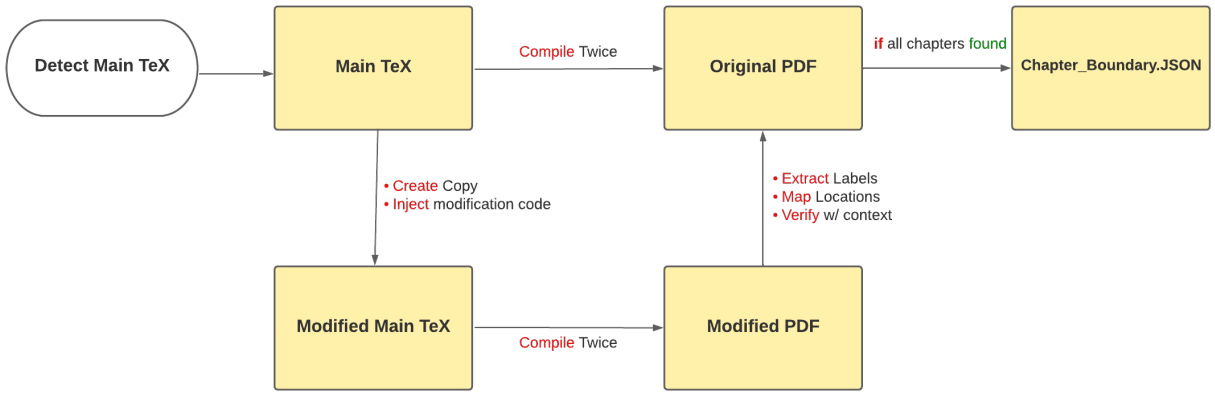


Figure 4.1: Chapter-Segmentation Pipeline

4.1 Overview

The Chapter-Segmentation Pipeline works by comparing the original PDF with a modified PDF representation of it. A code snippet is inserted at the beginning of a copy of the original TeX file. This wraps all chapter titles with special delimiters. Following this, we use a multi-step mapping process that is discussed in the following sections to map chapter boundaries. Figure 4.1 gives an overview of the pipeline and deliverable for each ETD. The

final deliverable after post-processing includes chapter boundary details. In particular, each chapter boundary is labelled as front-matter, chapters, or end-matter.

4.2 Motivation and Prior Attempts

Prior to our current Chapter-Segmentation Pipeline, we designed a separate pipeline that attempted to perform the same task of segmentation using a different methodology. This pipeline however failed to provide consistent results at scale. Our current method of segmentation works better, because it additionally verifies the start of a chapter using context text.

Working with LaTeX at scale can be tricky. Authors can use a wide variety of packages. They can also alter the behaviour of the typesetting environments. Further, LaTeX itself has progressively evolved throughout the years. The context text allows additional confidence when mapping chapter boundaries. Section 5.1 discusses the results of manually inspecting CSP generated chapter boundaries.

Another important benefit of the current CSP pipeline over the prior pipeline is its ability to differentiate between front-matter material, chapters, and end-matter material.

4.3 Generating Modified PDF

The steps taken to generate the modified PDF are similar to those for generating the original PDF, as discussed in Section 3.3.2. However, there is an additional step of inserting a modification script. The modified PDF is needed to determine the chapter boundaries for the original PDFs. Figure 4.2 shows a high-level overview towards using the modified and

original PDF to create a JSON file containing chapter boundary details. On the left is the modified PDF which is followed by the original PDF. The title of chapters and context (see additional details in Section 4.4.1) are extracted. Each chapter's title text is searched for within the original PDF and the correct instance is verified using the context. The final JSON deliverable comes with each chapter's title and page number.

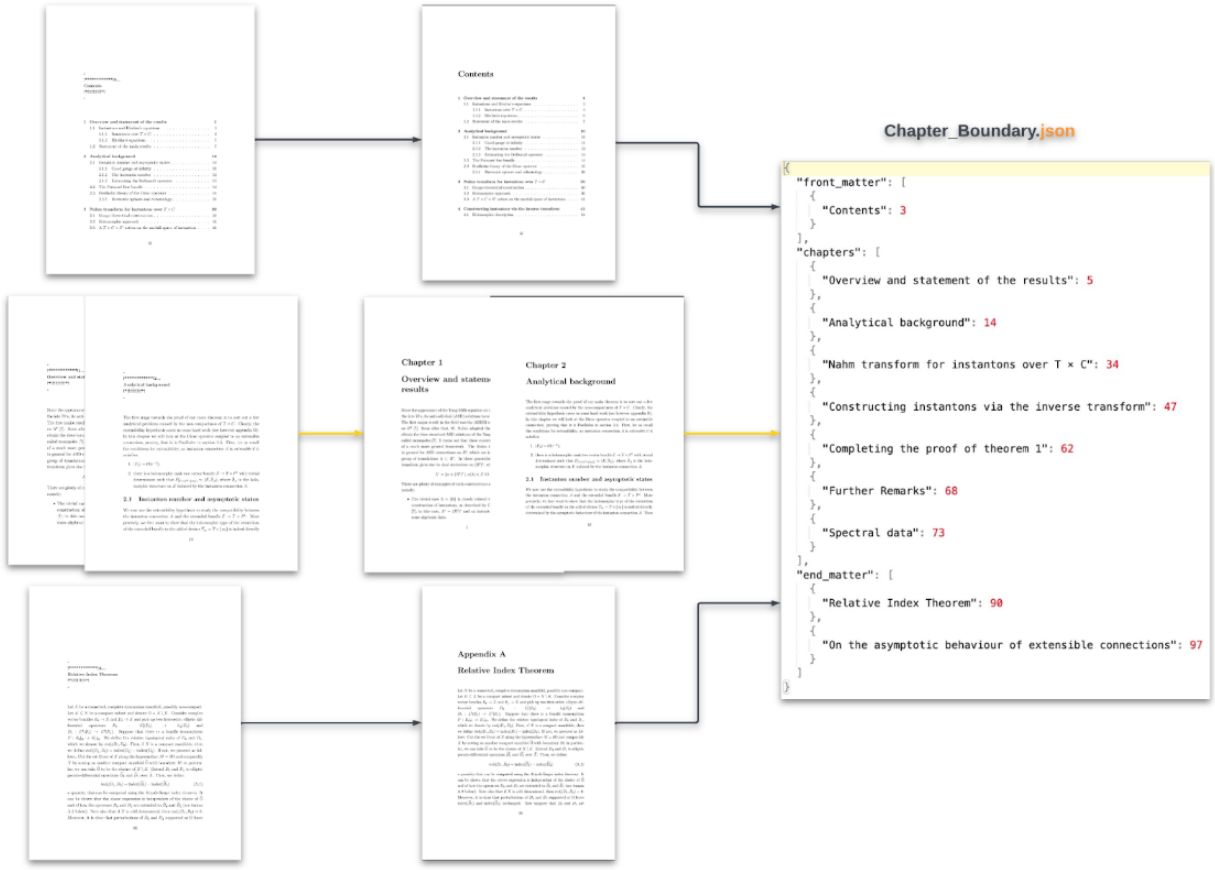


Figure 4.2: Generating chapter boundaries

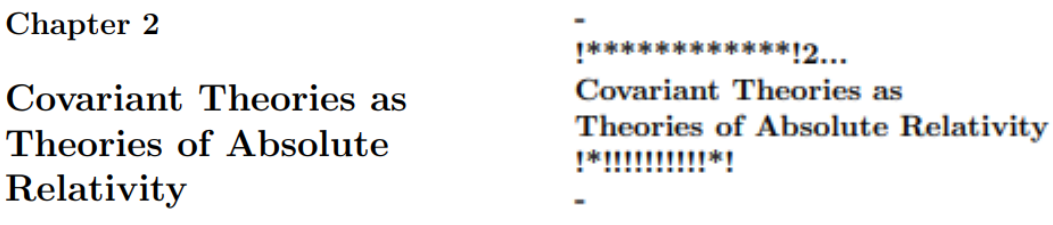
4.3.1 Modifying Main TeX File

The main TeX files were collected previously, when compiling the original PDF. Copies were made of the main TeX file, and the following lines of code were injected into the preamble section mentioned in Figure 3.3. This code will modify LaTeX's compile time behaviour to

wrap any instance of the **chapter** titles with special start and end delimiters. It also forces all chapter titles to include an enumeration before the title.

```
\usepackage{titlesec}
\usepackage{graphicx}
%Adding FORMATTING
\titleformat{\chapter}[display]
{\\bfseries}
{}
{0ex}
{- \\!************!\thechapter...\\}
[\\!*!!!!!!!!!!*!\\ - ]
```

The start delimiter **!************!** and end delimiter **!************!** can be seen in Figure 4.3. Within these tags it also includes the enumeration of 2 (in this case) before the title text. Such an enumeration is present for all chapter titles whether the original author chose to include them. The added chapter enumeration label allows to differentiate between front matter material, chapters, and end matter material. This is discussed further in Section 4.3.3.



(a) Original chapter title (b) Modified title

Figure 4.3: Modifying chapter titles

Figure 4.3 shows how chapter titles are modified. Keep in mind this adds additional text to the modified PDF, which makes 1-to-1 mapping between pages impossible.

4.3.2 Compiling Modified LaTeX Files

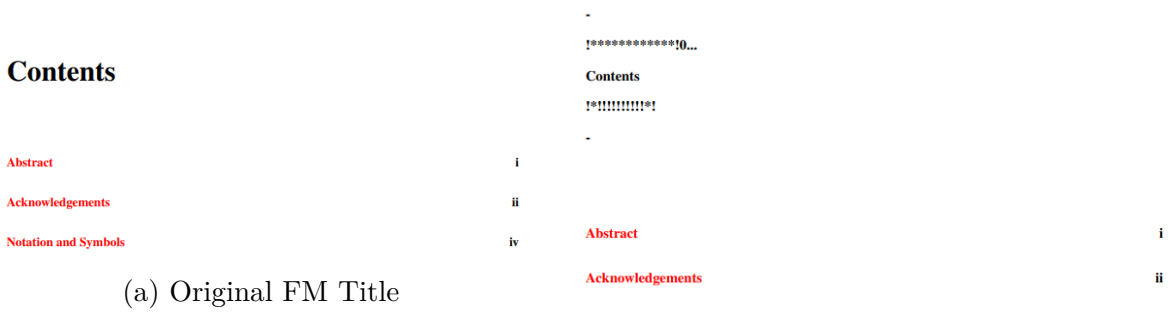
The modified LaTeX PDF was generated on the same system setup as the original. This was discussed in Section 3.3.2. The set of 1,872 source files that were able to generate the original PDF were processed using pdfLaTeX [37] on a *TeX-Live* [32] distribution. This was performed on a system using a Linux operating system. Before, we had experimented on a machine using Windows which provided us with inconsistent results. PDFs would generate for only a portion of the overall ETD before the TeX compiler would crash. Thanks to Dr. Heath's advice, this repository was regenerated on a Linux platform that removed the issue. From the set of 1,872 source files, 1,812 were able to generate the modified version of the PDF. In total 60 source files failed to generate the modified PDF. Table 4.1 shows a summary of the errors encountered.

Error	Description	Files
no legal \end found	Missing end } or \end of element	4
Over 100 Errors	Too many consecutive errors	15
Need more memory	TeX engine ran out memory	41

Table 4.1: Issues encountered while compiling modified PDF

4.3.3 Front Matter and End Matter Distinction

Arguably the biggest benefit of using this process is the additional chapter enumerations that are included within the modified chapter title. Including these will modify even front-matter and end-matter titles to include chapter enumerations, but since they are not actual chapters



(a) Original FM Title

(b) Modified FM Title

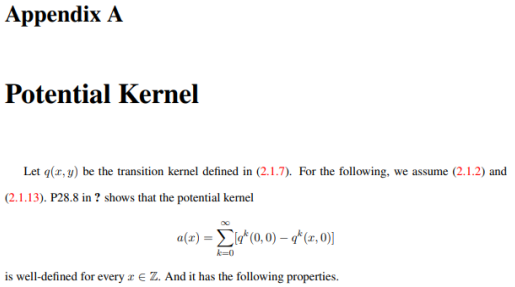
Figure 4.4: Modifying front-matter (FM) element titles

they behave differently, allowing users to distinctly label front-matter, chapter elements, and end-matter elements.

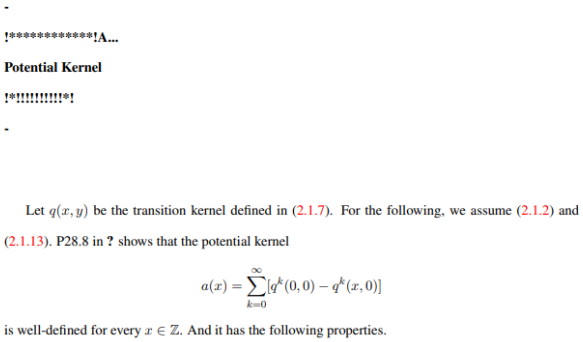
All front-matter element titles share an enumeration tag of 0 in the modified title. Figure 4.4 shows an example front-matter title and its modified representation. Figure 4.4b now includes a 0 enumeration alongside the title. This helps create distinctions between front-matter material and chapters. Documents can have varying front-matter material depending on the author's discretion.

As for elements that are end-matter such as appendices and bibliographies, there could be two different behaviours possible. Figure 4.5 displays the first behaviour. End-matter element titles are modified to include lettered enumerations. In the example shown in Figure 4.5, the letter **A** is added to the title and repeated till the last end-matter title.

The second behaviour that modified end matter titles can show are repeated enumeration of the last actual chapter's enumeration. Figure 4.6 displays one such example. Again, these are all dependent upon the author's discretion when writing documents using LaTeX. **Note:** Our modification pipeline can only capture front-matter and end-matter material if it has been declared using LaTeX's chapter command. The capture accuracy of each distinction is discussed in Section 5.1.

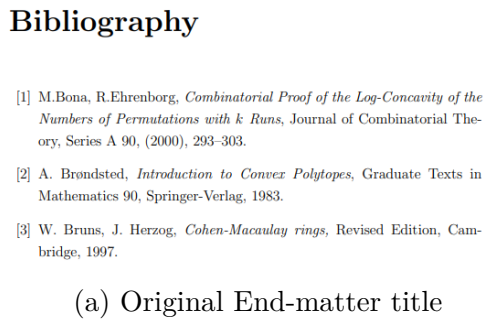


(a) Original EM title



(b) Modified EM title

Figure 4.5: Modifying end-matter (EM) element titles | **Version 1**



(a) Original End-matter title

(b) Modified End-matter title

Figure 4.6: Modifying end-matter (EM) element titles | **Version 2**

The actual chapters in between will follow an enumeration ranging from 1 to **n**. Using these simple heuristics it was possible to collect chapter boundaries on all ETDs and include the type of each chapter.

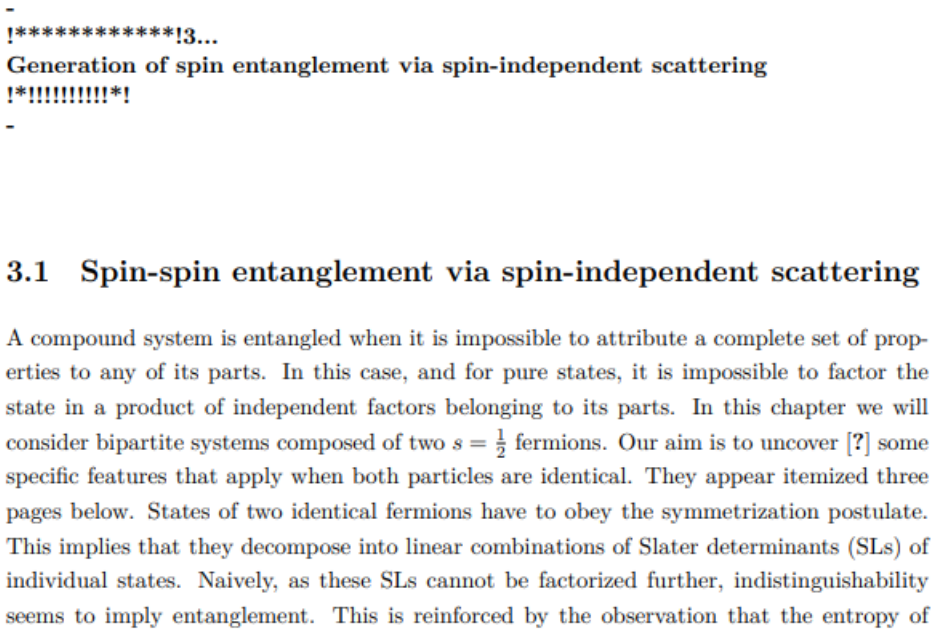
4.4 Title and Context

This section discusses preprocessing and extraction of chapter titles and context. As the delimiters are added onto the modified PDF the length of the PDF text slightly increases.

This does not allow for consistent 1-1 mapping between the two PDFs. The size of the title text also creates discrepancies in words that are split over lines or new pages. This difference is also observed within chapter titles. A small but crucial step in preprocessing is to remove the hyphen between any words that are split between lines. **Note:** Words that normally contained hyphens are not corrected. Using a Python package **PyMuPDF** [4], the text from both PDFs was extracted.

4.4.1 Extraction

Python [41] is used to search the text of the modified PDF for all start delimiters. All text between the start and end delimiters is extracted along with the chapter enumeration. While extracting the title of the text, additional text immediately following the title text is also extracted as ground truth context. As an example, Figure 4.7 shows a title and the text following it. The length of extracted ground truth context is discussed in Section 4.4.2.


-
!**********!3...
Generation of spin entanglement via spin-independent scattering
!*!!!!!!!!!!*!
-

3.1 Spin-spin entanglement via spin-independent scattering

A compound system is entangled when it is impossible to attribute a complete set of properties to any of its parts. In this case, and for pure states, it is impossible to factor the state in a product of independent factors belonging to its parts. In this chapter we will consider bipartite systems composed of two $s = \frac{1}{2}$ fermions. Our aim is to uncover [?] some specific features that apply when both particles are identical. They appear itemized three pages below. States of two identical fermions have to obey the symmetrization postulate. This implies that they decompose into linear combinations of Slater determinants (SLs) of individual states. Naively, as these SLs cannot be factorized further, indistinguishability seems to imply entanglement. This is reinforced by the observation that the entropy of

Figure 4.7: Chapter titles and context

The enumeration is separated from the title to categorize the type of chapter. The chapter titles and ground truth context are collected and organized in the order they are found within the modified PDF. While the length of the PDF may be altered by applied modifications, the ordering of chapters is preserved between the original and modified PDF. Figure 4.8 shows an organised set of titles from a sample ETD.

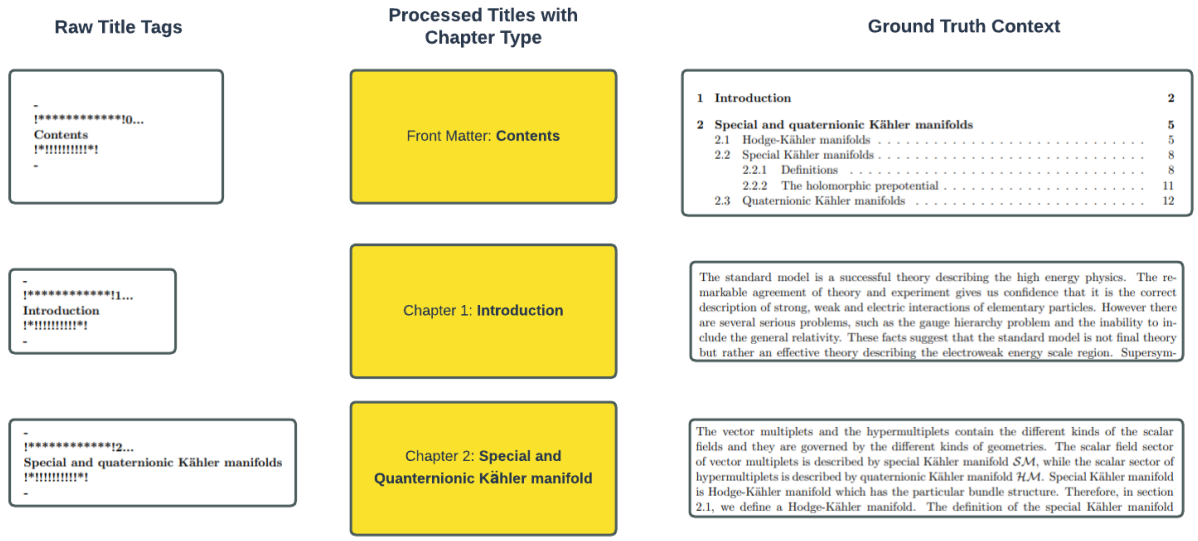
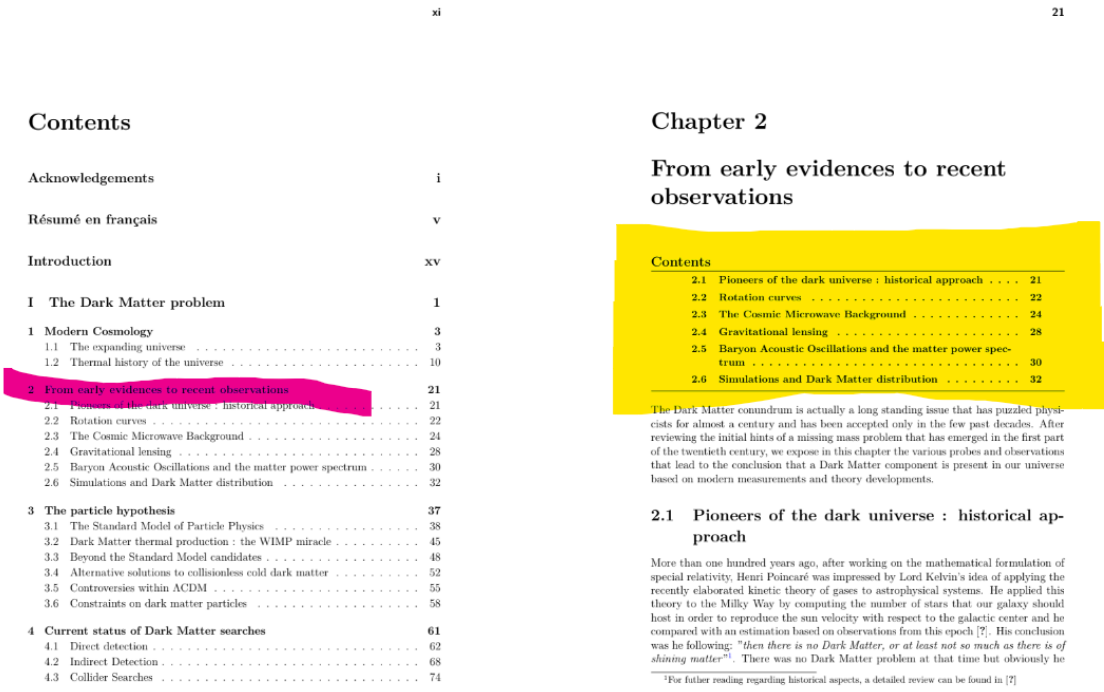


Figure 4.8: Organising chapter titles and ground truth context

Additionally, Figure 4.8 also shows processing of chapter enumerations to qualify chapter types. The contents chapter is qualified as front-matter, Introduction is qualified as Chapter 1, and so on. The title text and context text are then used for mapping, which we discuss in Section 4.5.

4.4.2 Context Length

The number of characters to choose to include in the context extracted following the title was difficult to determine. There is a difference in semantics between front-matter material, chapters, and end-matter material. If the context length is too short, it may lead to a faulty



xi

Contents

21

Chapter 2

From early evidences to recent observations

Contents

The Dark Matter conundrum is actually a long standing issue that has puzzled physicists for almost a century and has been accepted only in the few past decades. After reviewing the initial hints of a missing mass problem that has emerged in the first part of the twentieth century, we expose in this chapter the various probes and observations that lead to the conclusion that a Dark Matter component is present in our universe based on modern measurements and theory developments.

2.1 Pioneers of the dark universe : historical approach

More than one hundred years ago, after working on the mathematical formulation of special relativity, Henri Poincaré was impressed by Lord Kelvin's idea of applying the recently elaborated kinetic theory of gases to astrophysical systems. He applied this theory to the Milky Way by computing the number of stars that our galaxy should host in order to reproduce the sun velocity with respect to the galactic center and he compared with an estimation based on observations from this epoch [?]. His conclusion was he following: "*then there is no Dark Matter, or at least not so much as there is of shining matter*"[1]. There was no Dark Matter problem at that time but obviously he

[1]For futher reading regarding historical aspects, a detailed review can be found in [?]

Figure 4.9: Special ETD format can make short context ambiguous

mapping that refers to the wrong pages in the original PDF. To demonstrate this, Figure 4.9 shows an example thesis with a special format. Each chapter begins with an outline of the sections within it. This creates confusion during the mapping process as the title text appears to have the same context at two different locations within the PDF, i.e., one within the table of contents and the latter at the actual chapter start. In this case, too short of a context could lead to incorrect mapping.

There were also issues with having the context be too long. Too long of a context incorrectly assigned the label for the start of a chapter to a page **before** the actual chapter start. Sometimes authors will refer to the following chapter at the end of the prior chapter. Figure 4.10 shows one such example. The chapter title *Communication Complexity* is referred to

on the prior page. If the context length was long there could be a high amount of overlap in context.

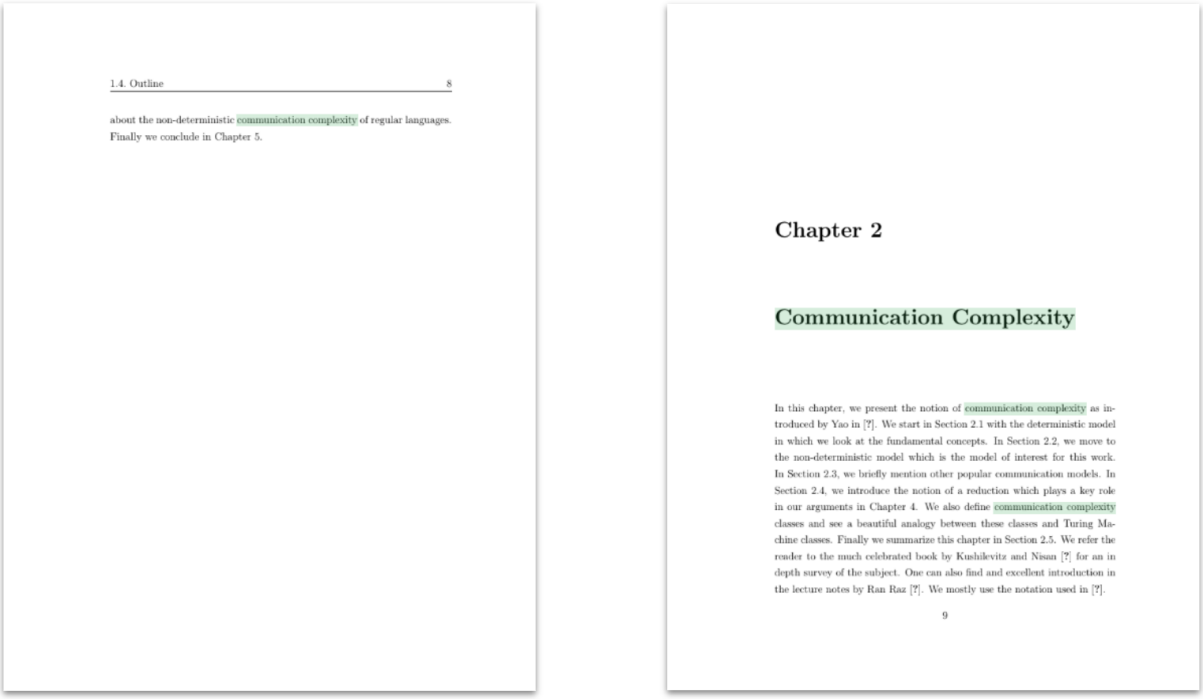


Figure 4.10: Consecutive pages in an ETD

Depending upon the case, this could result in an incorrect decision for a page before the actual chapter start. **Note:** Context is extracted across pages. Some ETDs do not start chapters on new pages.

Issues with larger context were observed less in comparison to the issues with shorter context. To deal with these issues, we decided to extract the context dynamically.

Dynamic Length of Context

To deal with the issues encountered above, the Chapter-Segmentation Pipeline extracts the following context of chapters dynamically. To perform this, a set of 20 ETDs where the CSP generated correct chapter boundaries, was inspected for their context. We calculated that the percentage of alphanumeric characters within the total context length was roughly **70%**. This was to deal with conflicts that could arise with title instances found in sections such as table of contents. **Note:** We conducted a study to potentially qualify the table of contents based on the extracted title text. This turned out to not be as trivial to qualify based on terms. We found a larger then expected variation in the way it can be named including terms from other languages.

Front-Matter or End-Matter Elements

For elements in front-matter or end-matter, static context of 400 characters in length was extracted. Roughly it would have the same length as the lorem ipsum passage shown below.

> Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt...

Actual Chapters

For actual chapters a static context of 400 characters is first extracted and compared with the 70% threshold of alphanumeric characters. It grows in length until the alphanumeric threshold is met. It would not exceed 800 characters to avoid having too long a context.

4.5 Mapping Start of Chapters

The Chapter-Segmentation Pipeline maps the collected chapter titles by searching for the instance of each title in the order that it collected them. Pages prior to the last found chapter start are not considered while searching for the next chapter title. It will however search for chapter titles on the same page and beyond. There are ETDs in our collection that are 30 pages or under, with short chapters.

Figure 4.11: Page character index-range

```
{
  0: (0, 250),
  1: (250, 254),
  2: (254, 1725),
  3: (1725, 2498),
  4: (2498, 3798),
  ...
}
```

It validates each chapter label using the context collected for each. As aforementioned, the context can span over a page break into the next page.

For this purpose a dictionary is created for the original PDF. It contains the progressive sum of the total characters in each page. The original text is extracted using **PyMuPDF** [4] as one long text.

4.5.1 Search and Validation

The full text of the original PDF is searched for a given chapter title and the found instances are collected. This implies there could be multiple found instances of each chapter title as shown in Figure 4.12. The title of a chapter can easily be referred to in other places.

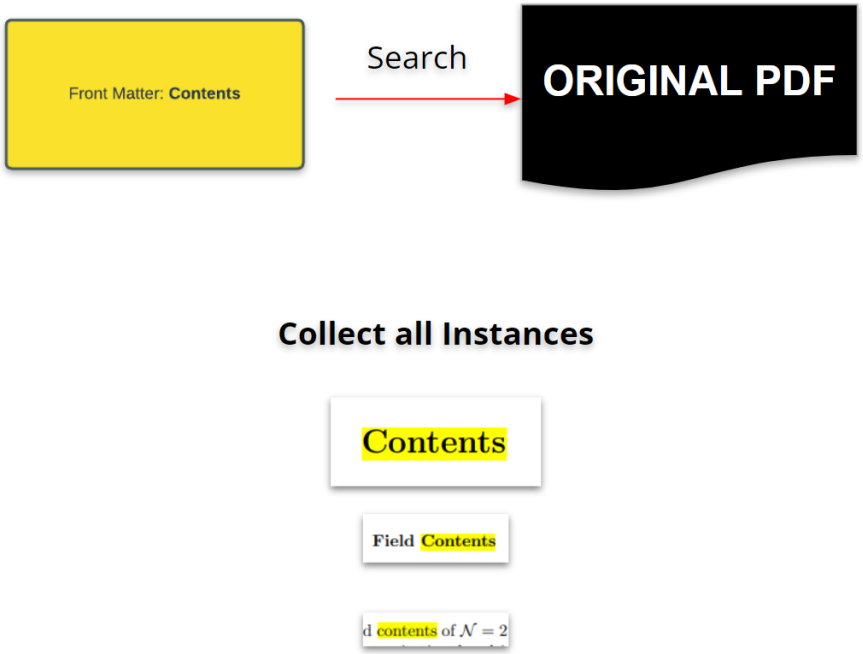


Figure 4.12: Collecting chapter title instances

For each occurrence, additional text is extracted that has the same length as the ground truth context. The **second** step involves comparing the ground truth **context text** with the following text of each occurrence.

The context is compared using cosine similarities between the two texts. The instance with the highest similarity is chosen as the correct instance of the chapter label as shown in Figure 4.13. If the highest similarity is less then 80%, the CSP will fail in the segmentation, as confidence in the context is not sufficient.

4.5.2 Final Segmentation Results

In total, we had 1,812 ETDs that were able to generate both the original and modified PDF. This set of ETDs was inserted into the Chapter-Segmentation Pipeline which was able to generate chapter boundary details for **1,459** ETDs using source files. A JSON with the

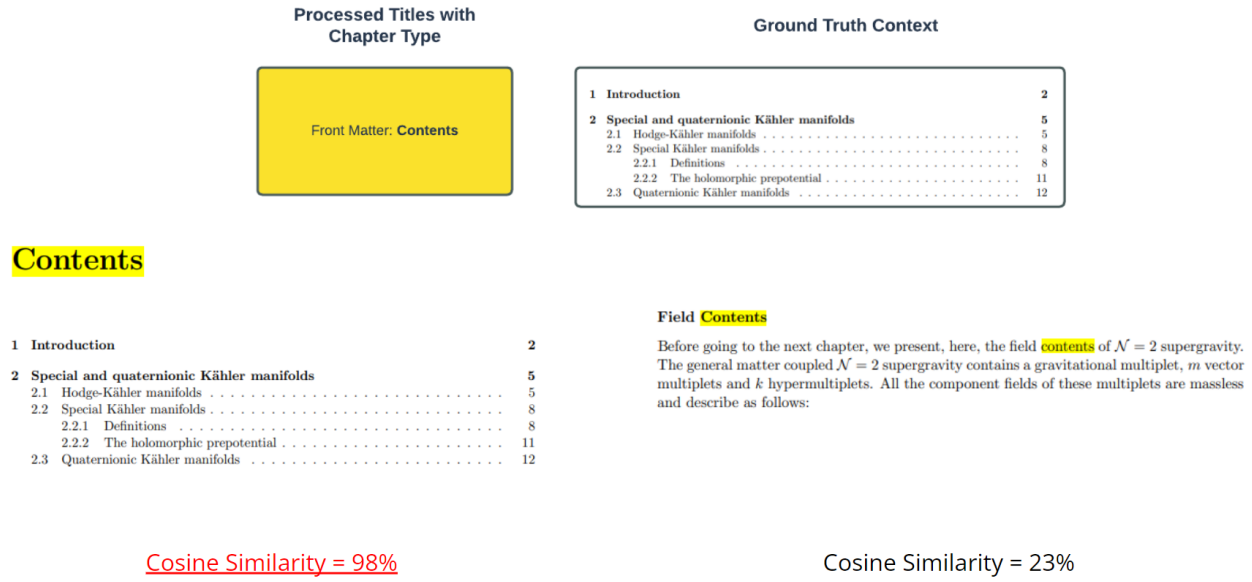
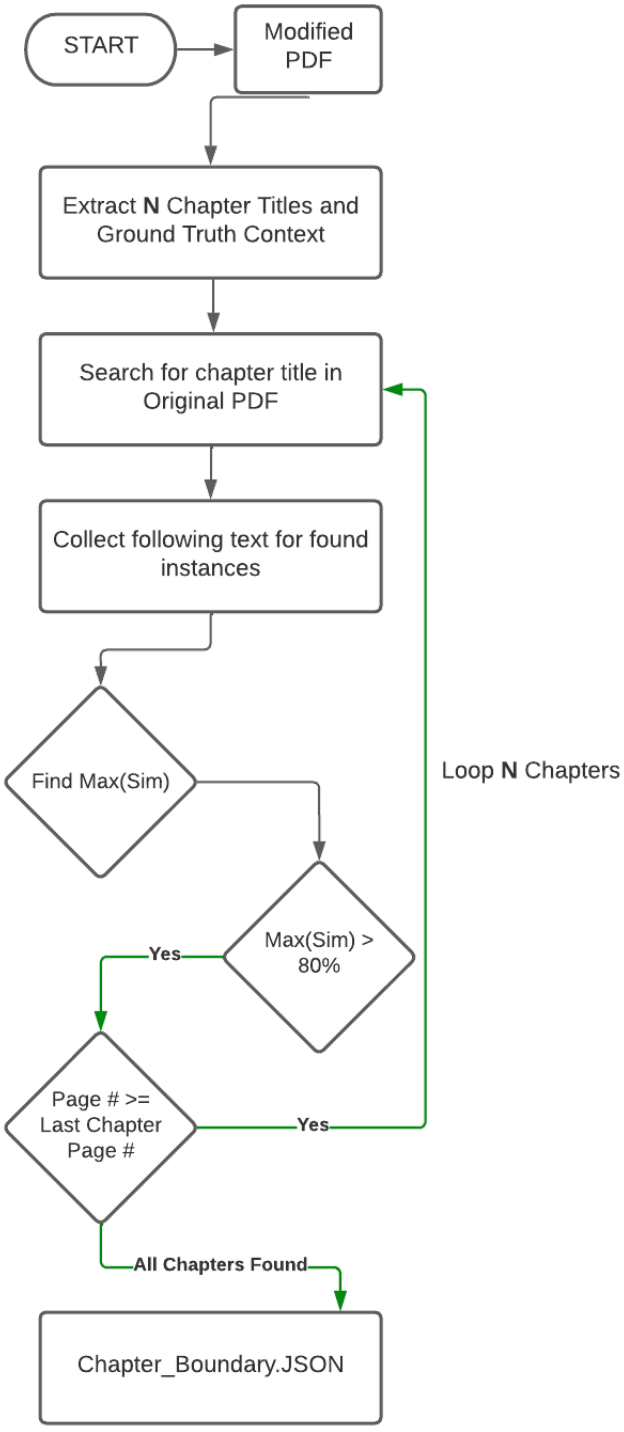


Figure 4.13: Verifying correct instance using context

collected chapter titles and the corresponding page numbers in the original PDF is created for each ETD. This JSON is shown in Figure 4.2.

The algorithm box in Figure 4.14 shows the step-by-step processing the CSP takes to generate chapter boundaries of a single ETD. The following chapter discusses the Chapter-Segmented ETD data set created using this pipeline. Note: If either condition marked in green is false, the CSP halts segmentation and returns void for an ETD.

Figure 4.14: Chapter-Segmentation Pipeline algorithm flowchart



Chapter 5

Chapter Segmented Data Sets

This chapter discusses the data set generated by the Chapter-Segmentation Pipeline (CSP). This data set was also used to train deep learning models that are discussed in the following chapter. Following this, it discusses a manually labelled data set. This data set is relatively small containing 150 ETD(s) with their chapter boundaries. The data set was used for the purposes of evaluating the deep learning models and comparison with other segmentation tools.

5.1 Manual Inspection

To verify the results of the Chapter-Segmentation Pipeline, numerous criteria were formulated. For each criteria, the outliers (top 3%) of papers were collected, which equated to 46 ETDs for each criterion.

1. High number of chapters: Above 17 chapters.
2. Low number of chapters: Between 2 and 5 chapters.
3. Short documents: Less than 37 pages in length.
4. Long documents: Above 254 pages in length.
5. Last chapter is unusually long: Last chapter is 27 pages or longer.

6. High number of chapters when compared to document length.

7. Low number of chapters, but long documents.

8. High number of chapters for long documents.

Among the set of 46 ETDs collected for each criterion, 10 ETDs were randomly sampled and inspected. In total this was **80** ETDs. If there were any ETDs with overlap with other criteria, another random sampling was performed to select its replacement in the same set of outliers.

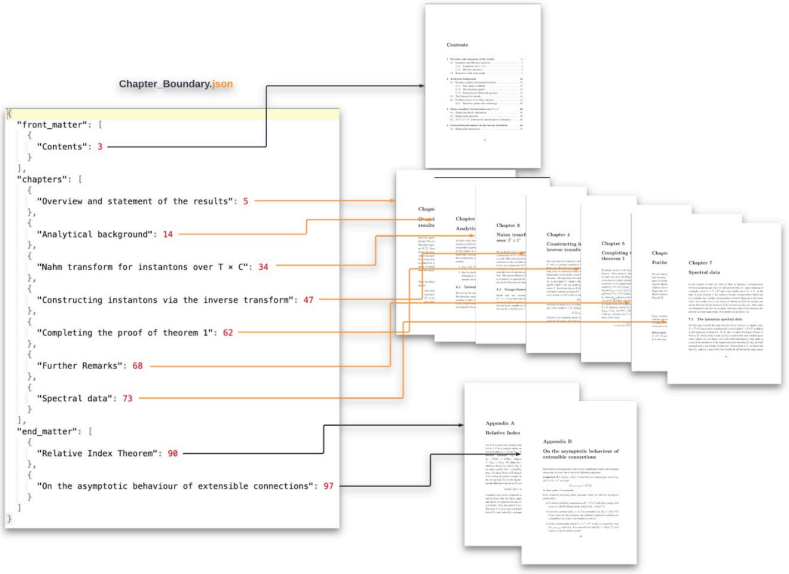


Figure 5.1: Validating generated page numbers

For each of these ETDs the generated chapter boundaries were inspected by manually verifying each auto-generated label as shown in Figure 5.1. No errors were found by manually inspecting the page numbers of each of these auto-generated labels.

The last test validated the page numbers of the auto-generated labels. Additionally, we verified the *capture accuracy*. This relates to the number of chapters within front-matter, chapters, and end-matter, as shown in Figure 5.2.

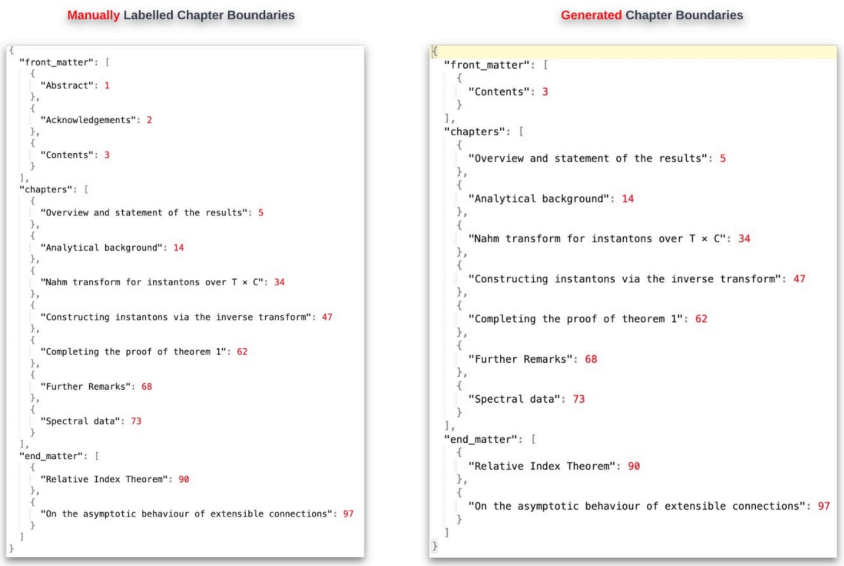


Figure 5.2: Comparing manually labelled vs. auto-labelled

This was performed by extracting a set of 35 ETDs that the CSP had auto-generated labels for. Another set of chapter boundaries were manually labelled for comparison. The results of comparing the two JSONs are shown in Table 5.1.

Chapter-Segmentation Pipeline chapter accuracy			
Chapter type	Missing **1** Chapter	Missing **2** Chapters	Missing **3** Chapters
Front-matter	8	10	2
Chapters	0	0	0
End-matter	4	0	0

Table 5.1: Inspecting CSP capture accuracy

Table 5.1 displays the capture accuracy of the different types of chapters between front-matter, chapters, and, end-matters. The CSP shows a perfect capture accuracy for actual chapters, but shows less capture accuracy for front-matter and end-matter material. As mentioned earlier in Section 4.3.3, elements in front-matter or end-matter can be initialized using special packages. Our pipeline needs the chapters to be initialized using LaTeX's

chapter command. We manually labelled dedications, acknowledgements, or dedications as chapters but these can often be initialized using other commands. This is reflected above as the CSP fails to capture often between 1 or 2 chapters within front-matter material. Since our research focus has been capturing chapters, we proceed with these results.

5.2 Collection Overview

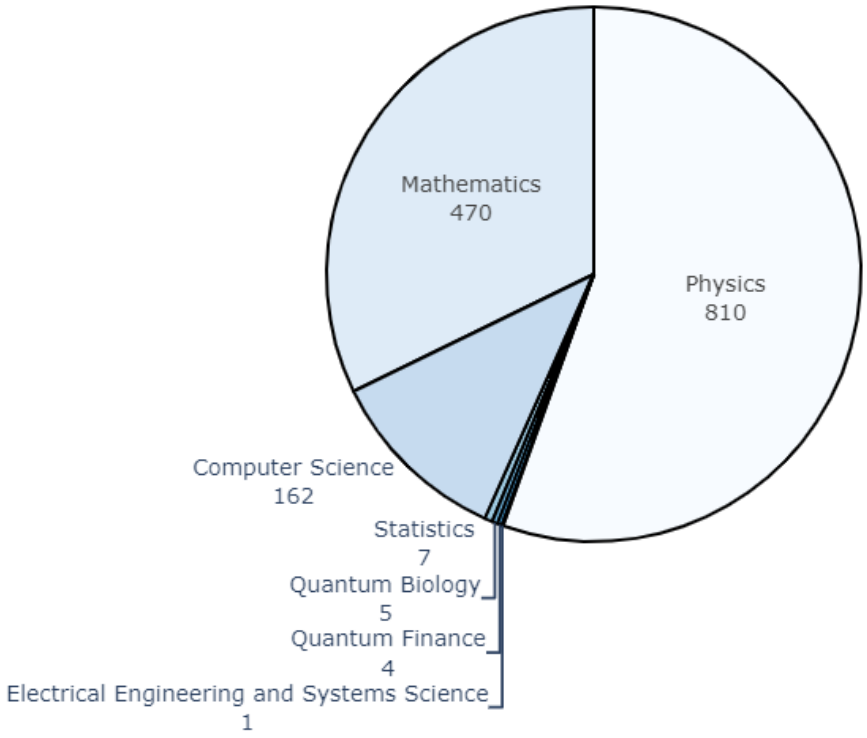
This section discusses various characteristics regarding the Chapter-Segmented ETD (CSE) data set that was generated as a result of the segmentation pipeline. In total the CSE data set contains 1,459 ETDs. The pie chart in Figure 5.3 shows the primary category distribution in the generated set. The collection has a similar distribution of categories as arXiv's overall category distribution.

Each ETD's folder retains the original arXiv directory structure and files as shown in Figure 3.2, but with new additional files. The important new files are the chapter boundary JSON files, original PDFs, and modified PDFs. Each PDF (including the original) has been generated by compiling provided source LaTeX files using pdfLaTeX [37].

The collection contains ETDs from 2007 to 2022. Figure 5.4 shows the number of documents for each year in our repository. These are not by publication date, rather by last updated dates as users on arXiv can update submissions. arXiv still retains previous versions after a user updates their document(s).

5.2.1 Document Statistics

This section discusses various statistics regarding document properties and generated chapter boundary details. For instance, the average number of chapters for documents in the repos-



Category	Documents	Ratio
Physics	810	55.6%
Mathematics	470	32.2%
Computer Science	162	11.1%
Statistics	7	0.48%
Electrical Engineering and Systems Science	1	0.068%
Quantum Biology	5	0.343%
Quantum Finance	4	0.274%

Figure 5.3: CSE data set: Document distribution for arXiv primary categories

itory is **6** chapters. Figure 5.5 shows box plot distributions of chapter-starts in front-matter material, chapters, and end-matter material. The Y-axis shows the number of chapter-starts in each.

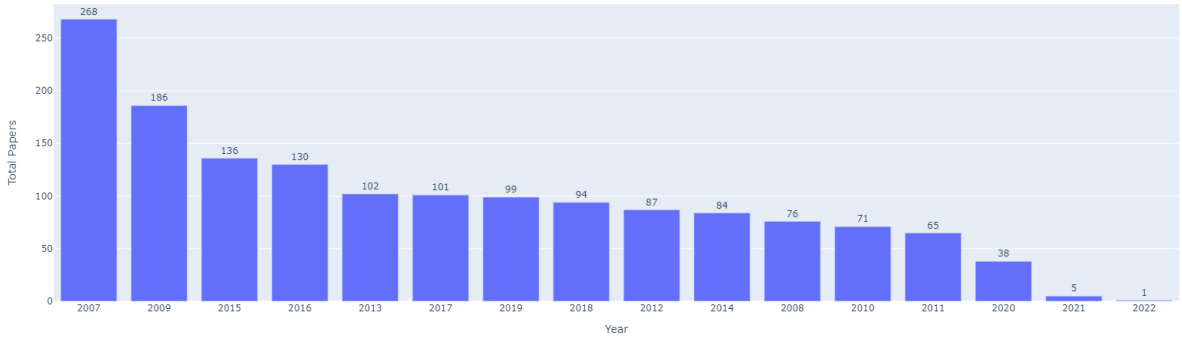


Figure 5.4: ETDs by last updated year

Figure 5.5: CSE data set: Comparison of chapter distributions

The maximum number of chapters in a single ETD is 23. The average length of chapters in this ETD is 7 pages which is relatively low in comparison to the global average of 19 pages. There is also an ETD with 19 elements as end-matter material. In this ETD, each chapter contains a citations page for references used in that specific chapter. Our pipeline qualifies each as end-matter material regardless of its positioning in the document. There are many other documents in our repository that have been written in a similar style as this one.

Table 5.2 shows a few important statistics regarding the overall ETD segmentation data set. The average length of front-matter elements is 3 pages. This makes sense however. Front-matter elements such as Table of Contents, Table of Figures, and Prefaces are generally 1-3

Characteristic	Values
Average no. of Chapters	6 chapters
Average no. of Pages	127 pages
Average Chapter Length	17 pages
Longest ETD	450 pages
Shortest ETD	6 pages
Highest No. of Chapters	20 chapters

Table 5.2: Chapter-Segmented ETD data set statistics

pages long. Further research was done to see if the length of an ETD has a higher correlation with the length of a chapter or the number of chapters. The scatter plot in Figure 5.6 shows the length of a chapter has a positive correlation with the total document length. The green line shows the correlation between total number of chapters and total document length, which shows little to no correlation. This shows longer ETDs generally have longer chapters, not more of them.

Figure 5.6: Document length vs. Avg. length of chapters

5.3 Manually Labelled Data Set

The documents that were successfully segmented using our second LaTeX modification pipeline discussed in Section 5.2 were chosen to train the deep learning models. Section

5.1 discusses the validation process for the overall data set generated through the segmentation pipeline. To properly evaluate the segmentation pipeline and deep learning model performance, a manually labelled data set of 150 ETDs was created. Chapter boundaries of each document were manually labelled. Chapter-starts are distinctly labelled between front-matter material, chapters, and end-matter material. This provides a reliable way to judge the performance of the deep learning models and compare the segmentation pipeline with other chapter segmentation tools such as GROBID (see Section 7.1).

5.3.1 Collecting ETDs

For the purpose of wider representation of ETDs, we collected ETDs from two separate sources. The first was arXiv, which has a limited number of disciplines (see Section 5.3). The latter were provided by Bipasha Banarjee and they were collected from Old Dominion University's (ODU) corpus of ETDs from around the USA. The ODU corpus contains a collection of some 500,000 ETDs. These were collected without any restrictions on discipline which allowed us to have a wider representation of ETD subjects.

Prior in Section 3.1.1, we mention the works of Bipasha Banarjee. She collected and manually verified 2,838 ETDs from arXiv. We randomly sampled the unique IDs in this subset (as they are verified as ETDs) and selected a total of 100 ETDs. This created an overlap of 29 ETDs with our overall segmentation data set (1,459 ETDs). During deep learning training stages, overlapping ETDs were filtered out of the training and validation sets to keep the testing data set unique. Thus, only 1,430 ETDs were in either the training or testing set. Table 5.3 displays the categories and total document count collected from arXiv.

Additionally, a total of 50 ETDs were collected from ODU's corpus of ETDs through random sampling. Refer to Table 5.4 for categories and document counts.

Discipline	No. of ETDs
Physics	60
Mathematics	31
Computer Science	9
Total	100

Table 5.3: ETDs collected from arXiv

Thus we could evaluate our deep learning models separately on the 100 ETDs from arXiv, and on the 50 from the ODU corpus. These results are discussed in Chapter 7.

5.3.2 Category Distribution and Document Statistics

Discipline	No. of ETDs
Mechanical Engineering	14
Electrical and Computer Engineering	10
Bio-engineering	6
Statistics	4
Environmental Engineering	4
Education	3
Physics	2
Computer Science	2
Aerospace Engineering	2
Architecture	2
Geography	2
Chemical Engineering	1
Total	50

Table 5.4: ETDs collected from ODU's corpus

Refer to Table 5.3 and Table 5.4 for primary category distributions of ETDs within our manually-labelled dataset.

Figure 5.7 displays the distribution of Chapters, Front-Matter element starts, and End-Matter element starts in the manually labelled data set. The average document length for

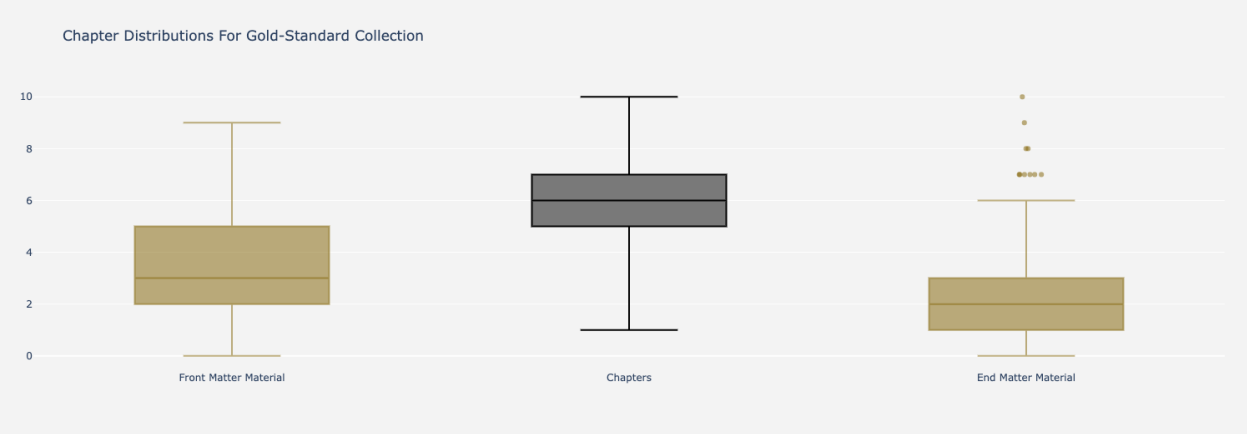


Figure 5.7: Comparison of chapter distribution

the collection of ETDs in the manually labelled data set is **142** pages containing on average **6** chapters. Table 5.5 displays various statistics regarding the collection.

Characteristic	**Values**
Mean Number of Pages	142 Pages
Mean Chapter Length	20 Pages
Mean Front-Matter	2 Pages
Mean End-Matter	11 Pages
Longest ETD	340 Pages
Shortest ETD	24 Pages

Table 5.5: Collection statistics for manually labelled data set

Figure 5.8 shows a correlation analysis between total page length and length of chapters. The dark blue regression line displays the correlation trend between ETD lengths and the length of chapters within. The green however displays the correlation between the total number of pages and the number of chapters. These results are consistent with results discussed in Figure 5.8 for our generated data set. The length of a document has a positive correlation with the length of its chapters. Thus, longer ETDs tend to have longer chapters instead of more chapters.

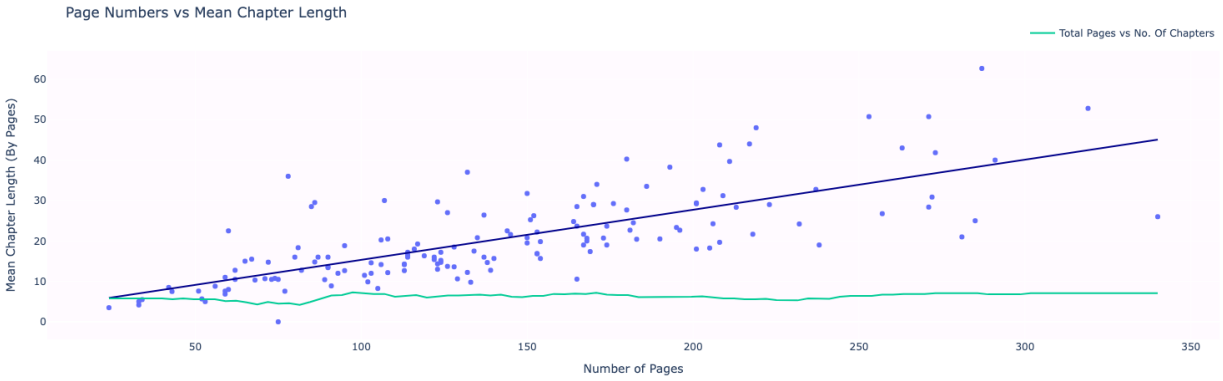


Figure 5.8: Manually labelled data set: Document length vs. Avg. length of chapters

Chapter 6

Deep Learning Models

This chapter discusses the machine learning and deep learning models that were trained using the Chapter-Segmented ETD data set. A description of the features is discussed first, including how page images and page text were preprocessed using embeddings. The chapter next discusses the design of deep learning models that were used. That is followed by a description of the training setup, including the input splits. Finally is an explanation of the evaluation metrics used to assess the models.

6.1 Overview of Models

The deep learning models were designed to learn page sequences by tagging chapter boundaries. The code for the deep learning models was inspired by the architecture of Part-of-Speech taggers, particularly by the works of Ben Trevett and Antonio Sejas for POS tagging using PyTorch [38]. Pages in each ETD were treated as sequences and each page was tagged with its respective distinction such as front matter section starts, chapter-starts, and end-matter section starts. This was possible as ETDs often organize material in each of the sections with similar contents. Sections such as the table of contents usually only exist in front-matter material, while appendix sections will be included in end-matter material. The deep learning models were tested as single input models (text or image) and combined input models (text and image) to make comparisons.

6.2 Feature Descriptions

For the deep learning models, each page is an element in a sequence tagged by its qualifying distinction. For each page, images (visual) and text (context) were considered to be valuable inputs to our models. This section discusses how image features and converted text were transformed into pretrained embeddings.

6.2.1 Page Image Feature Extraction

Transformations

Much of the work was inspired by alumnus Sampanna Kahu's efforts with figure extraction [16]. Since the Chapter-Segmentation Pipeline was able to only generate chapter boundaries for 1,459 documents, transformations were necessary. PyTorch [10][28] was used to implement image-based transformations using its transformation library which resulted in the following transformations. Note: Transformations did not alter or increase the size of the training set. Thus, each original page in the 1,459 documents was changed to a new page, by applying each of the three following random transformations, and then the new page was used in training of a deep learning image-based model.

- **RandomAffine Rotations (-5, 5)**: It is possible that scanned ETDs might have tilt in scanned pages. Adds slight random rotation to each image. See Figure 6.1.
- **GaussianBlur (sigma = 0.5):** A certain degree of sharpness can be lost while digitizing physical hard copies. Randomize resolutions of page images to allow smoothing.
- **RandomAutocontrast: 0.5:** Given an image this will randomize the contrast slightly on all given pixels.

- **RandomPerspective: 0.025:** While scanning hard-copies, it's possible ETDs might be stretched. This adds a slight degree of random yaw to each page image.

Feature Extractions

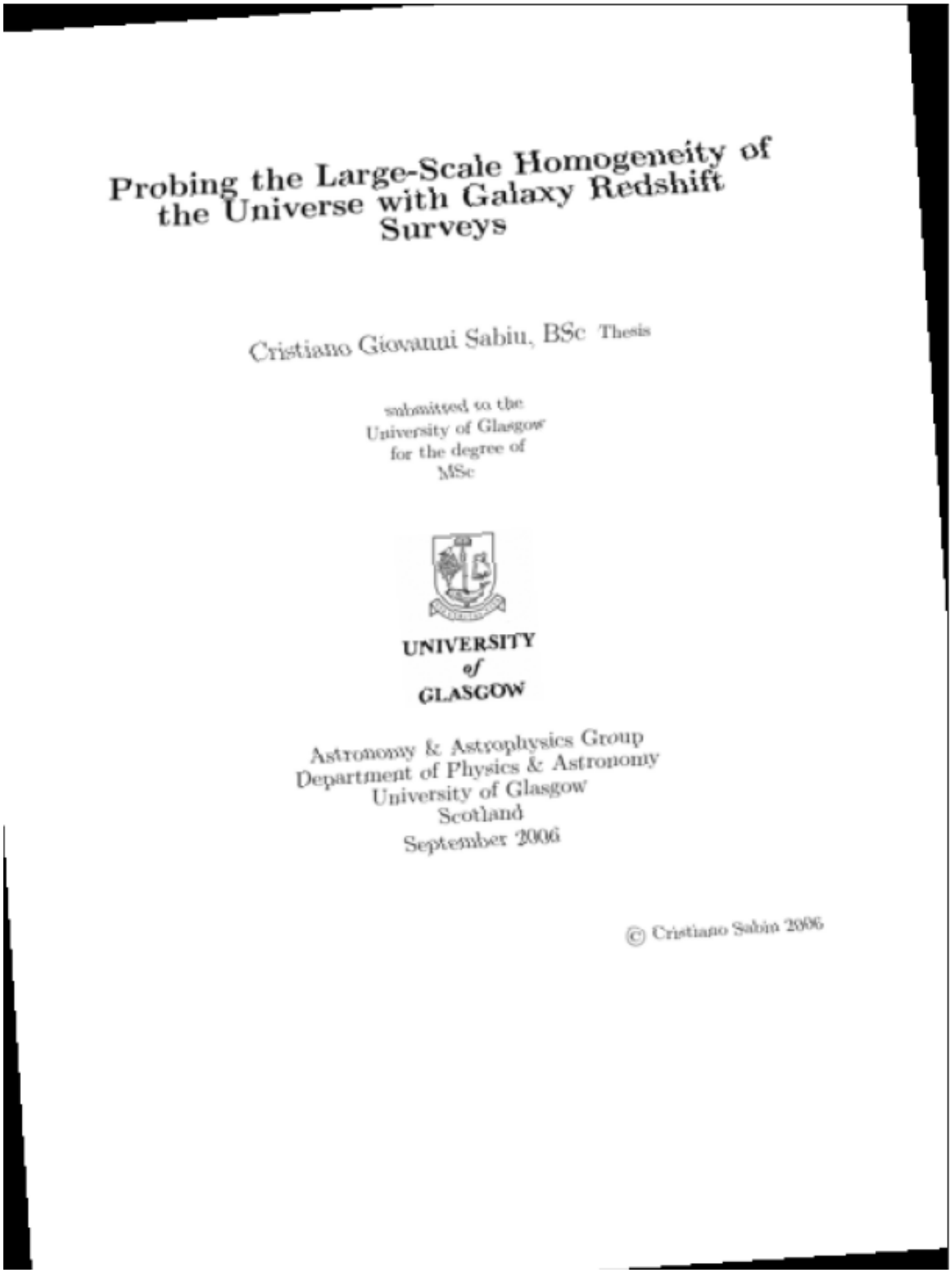
Image features were extracted using VGG16. VGG16 is a convolutional neural network that is 16 layers deep [35]. VGG16 has been trained on a varied set of images and has a rich feature representation for a wide range of images. VGG16 can also be trained for specific classification tasks. Since the work specifically requires ETD page images, VGG16 was fine-tuned for ETD page images. Generic feature images for page images were extracted from VGG16's penultimate layer, which is also known as the FC7 layer. See Figure 6.2.

6.2.2 Text Embeddings

The text was converted into embeddings using pretrained libraries of GloVe [31] and fastText [7] so as to compare the performance of each. These embeddings were further trained for this task as well. To reduce computational complexity, only the first 250 words of each page were used. Text was preprocessed using the following steps before converting into embeddings:

- Removal of new line characters.
- Removal of any punctuation.
- Tokenization using PyTorch's tokenizer [28].

Following tokenization, text was converted using the following different pretrained embeddings and used for separate experiments.



Probing the Large-Scale Homogeneity of the Universe with Galaxy Redshift Surveys

Cristiano Giovanni Sabiu, BSc Thesis

submitted to the
University of Glasgow
for the degree of
MSc

UNIVERSITY
of
GLASGOW

Astronomy & Astrophysics Group
Department of Physics & Astronomy
University of Glasgow
Scotland
September 2006

Figure 6.1: Sample page image transformation

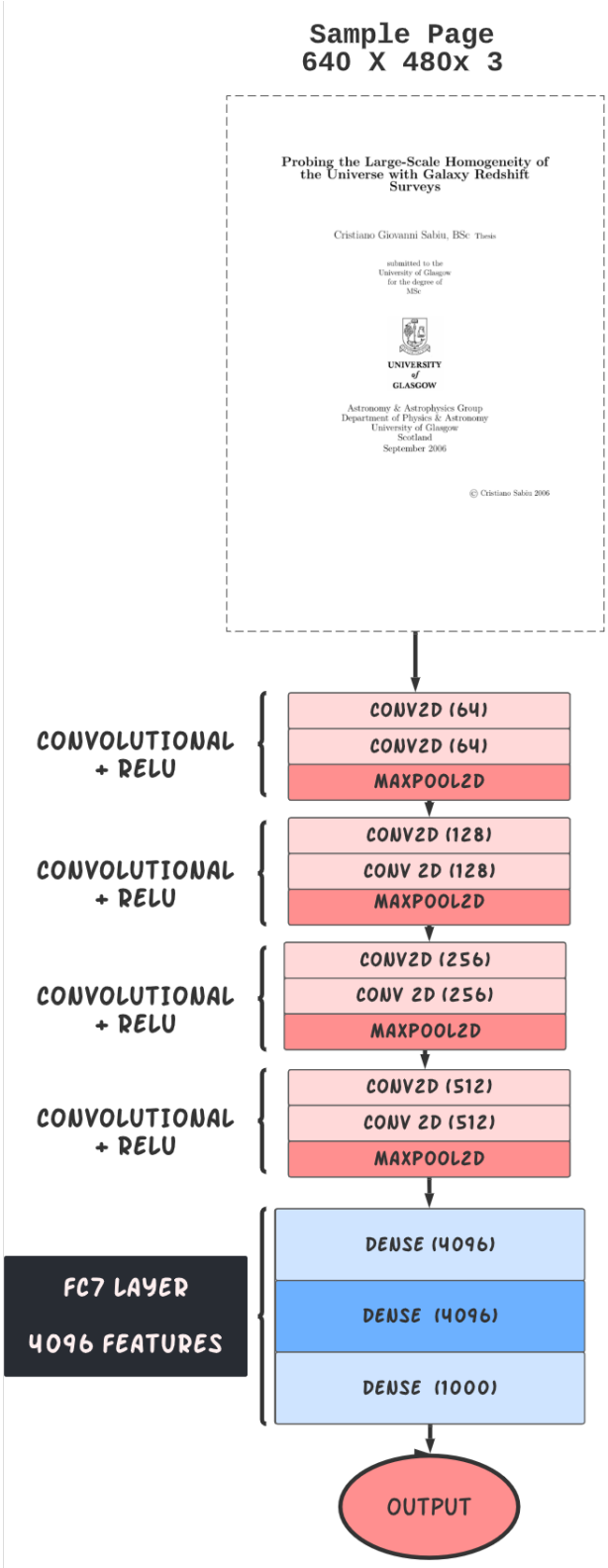


Figure 6.2: VGG16's FC7 layer for feature extraction

- **GloVe:** glove.6B.100d [31]
- **fastText:** cc.en.300.bin (converted to 100d) [7]

6.3 Single Input Models

6.3.1 Image-only Sequence Tagger

Figure 6.3 shows an overview of the first deep learning model's data flow. In this model, only the page images are being used as an input. All pages in ETDs are converted into page images as PNGs using a Python package pdf2image [5]. Then transformations were applied as in Section 6.2.1. These page images are organized into a sequence where each element is tagged with its sequence label. The data set was partitioned according to a standard **80/20** split, leading to training with 80% and validation with 20%. Before performing the split any ETDs that were included in the Manually Labelled data set were removed to avoid overlap (29 ETDS). Refer to Section 5.3.2 for more details on our Manually Labelled data set. The training set contained 1144 documents while the validation set contained 286 documents.

Refer to Figure 6.4 for a detailed description of the architecture of the Image-only model architecture. Pages are treated as images in a sequence similar to frames in a video. Features are extracted from the **pretrained** VGG16 which are inputs to the Bidirectional LSTM. The results of the model training are discussed in Section 7.2.

6.3.2 Text-only Sequence Tagger

The Text-only model has a similar design as the first deep learning model shown in Figure 6.3. This model embeds the text of each page into pretrained embeddings discussed in

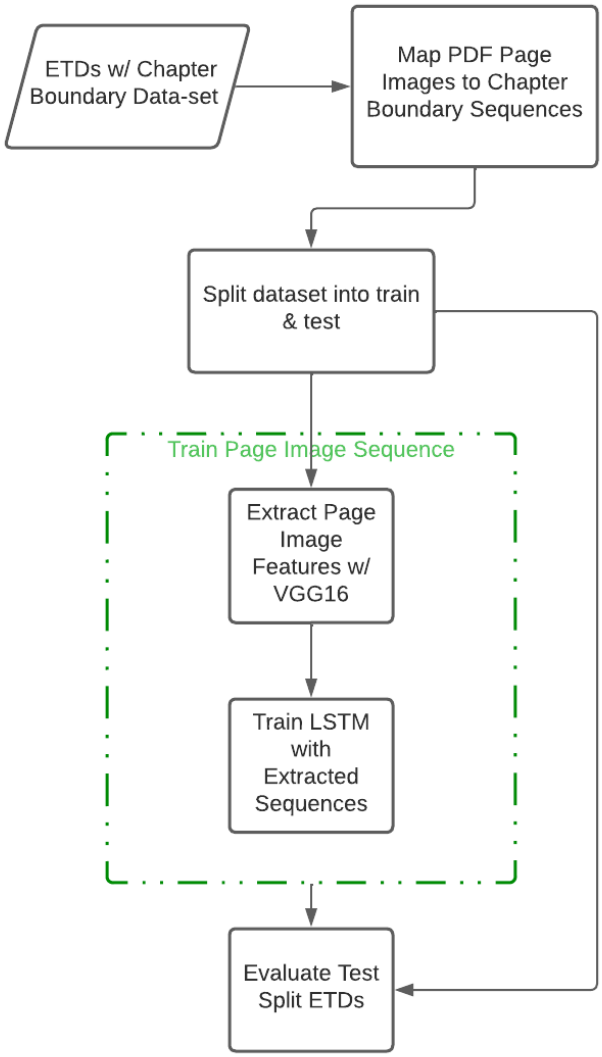


Figure 6.3: Image-only sequences: Data-flow diagram

Section 6.2.2. It follows this by extracting the features of the embedded text from each page using the CNN architecture of the VGG16. **Note:** The pretrained model weights of the VGG16 were not initialized. The features of each page text are inputted as a sequence in a Bidirectional LSTM. Refer to Figure 6.5 for the architecture of the Text-only model and Figure 6.6 for a data-flow diagram.

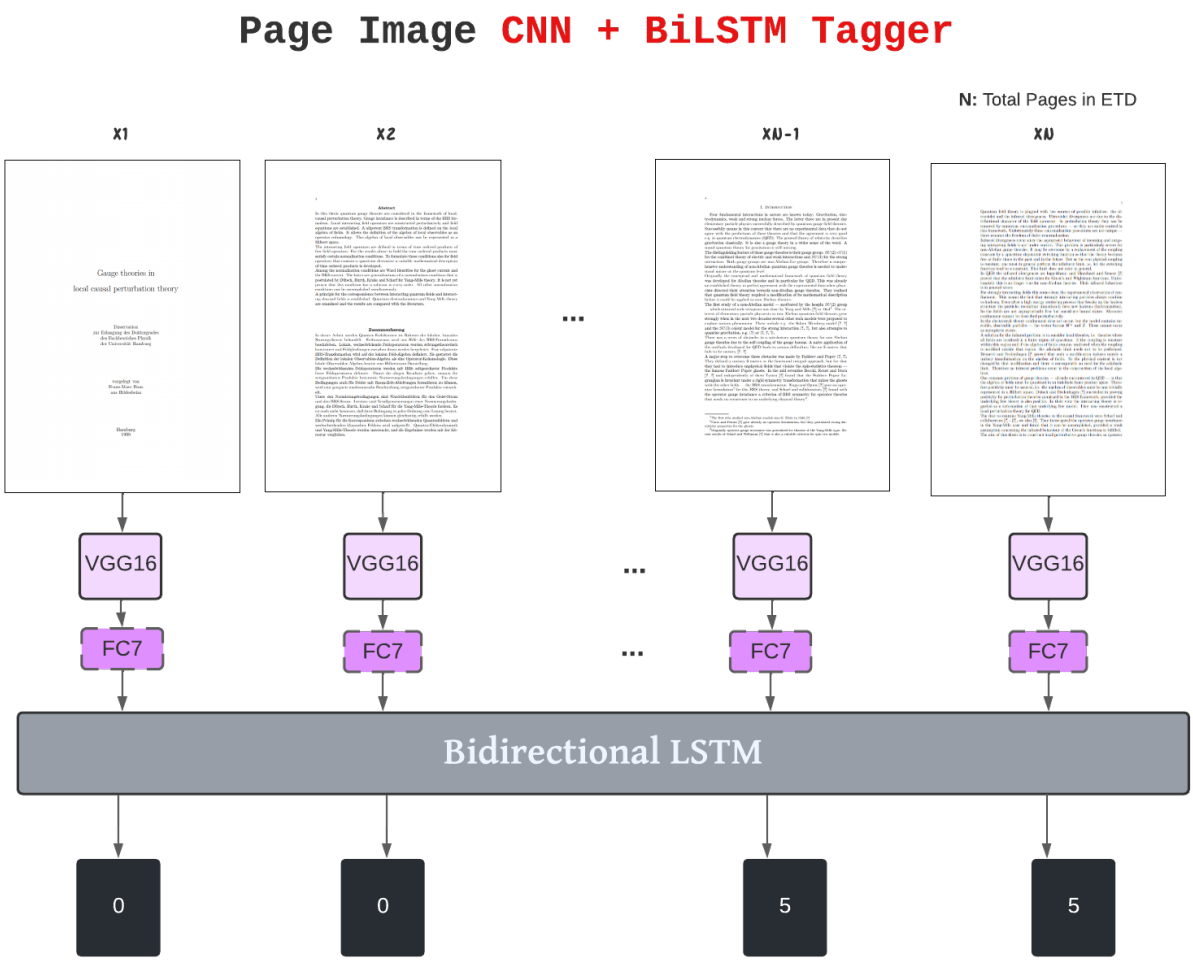


Figure 6.4: Image-only sequence tagger: Model architecture

6.4 Image and Text Sequence Tagger

The final model is our combined input model that includes each page's image and text features as sequence elements. The data flow diagram for this model is shown in Figure 6.7. Similar to the first data flow diagram, ETDs are converted into a sequence of PNG images of each page. Then transformations were applied as in Section 6.2.1. Features are extracted from each page image using VGG16. The text is embedded using various

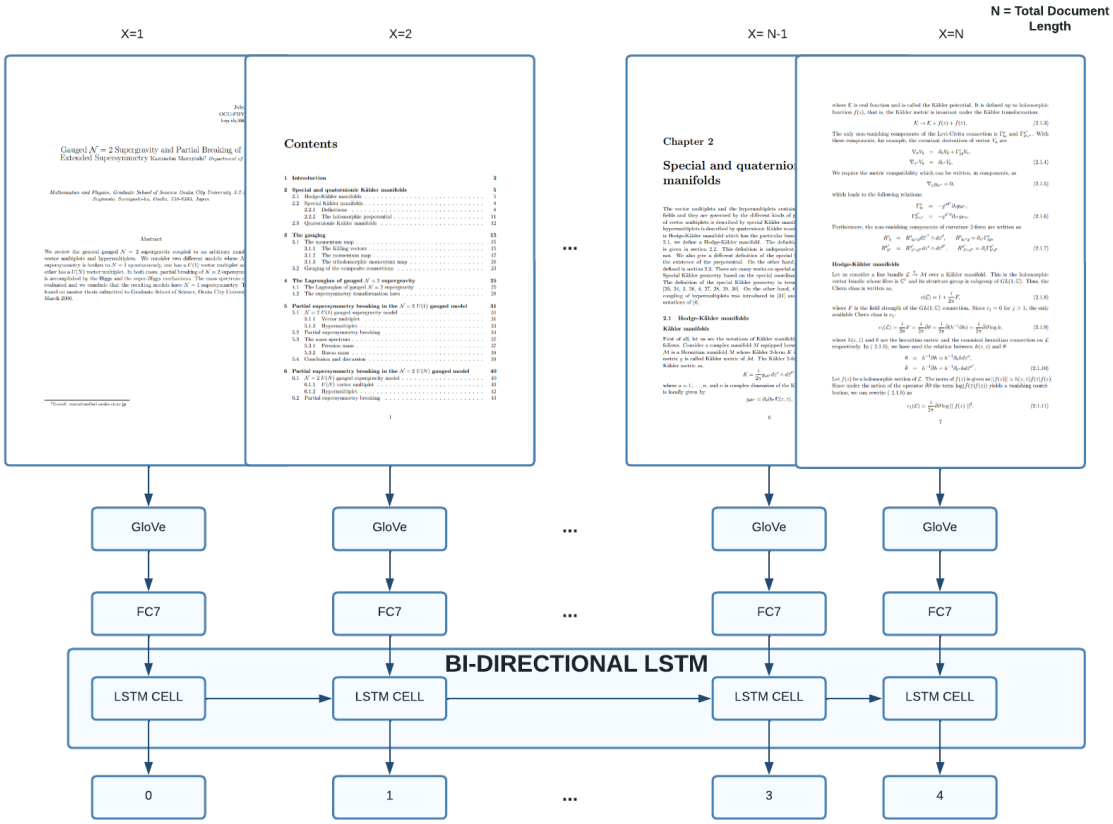


Figure 6.5: Text-only sequence tagger: Model architecture

embedding techniques as described in Section 6.2.2 and its features are extracted using a separate CNN. The image features and text features are then combined using a linear layer.

The design of the combined model is displayed in Figure 6.8. This model builds upon the single input models but includes both inputs. Model performance is discussed and compared further in Chapter 7.

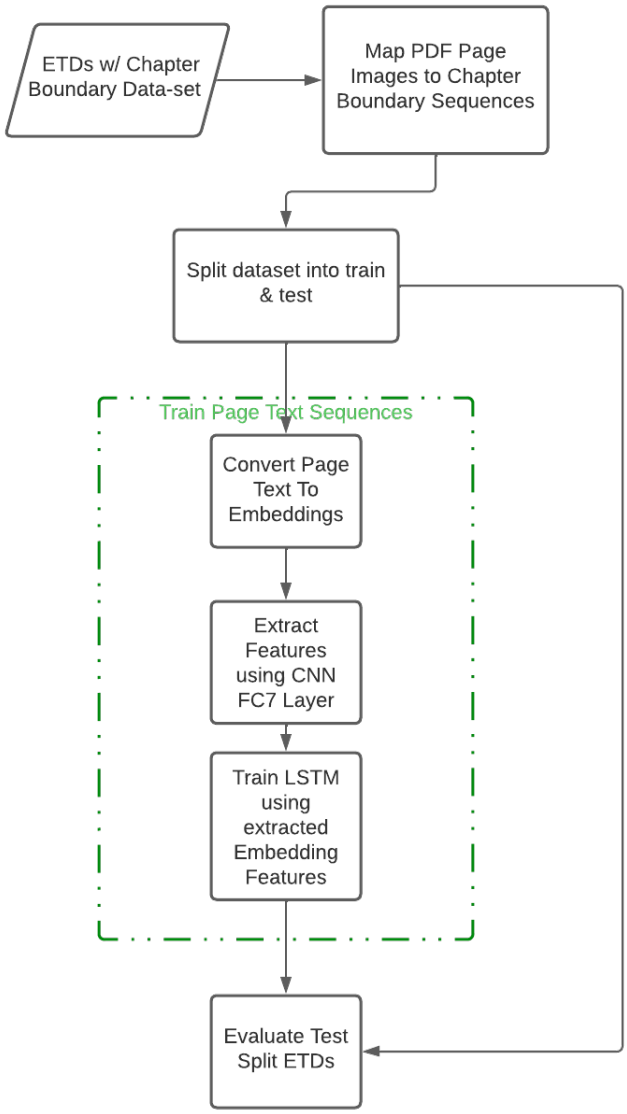


Figure 6.6: Text-only sequences: Data-flow diagram

6.5 Evaluation Metrics

Since a multi-label classification task is being performed on a sequence, the performance of the models using accuracy, precision, recall, and F1 score evaluation metrics is discussed. Outcome labels include six possible labels, i.e., front-matter element start, front matter in-element page, chapter-start, in-chapter page, end-matter element start, and end-matter

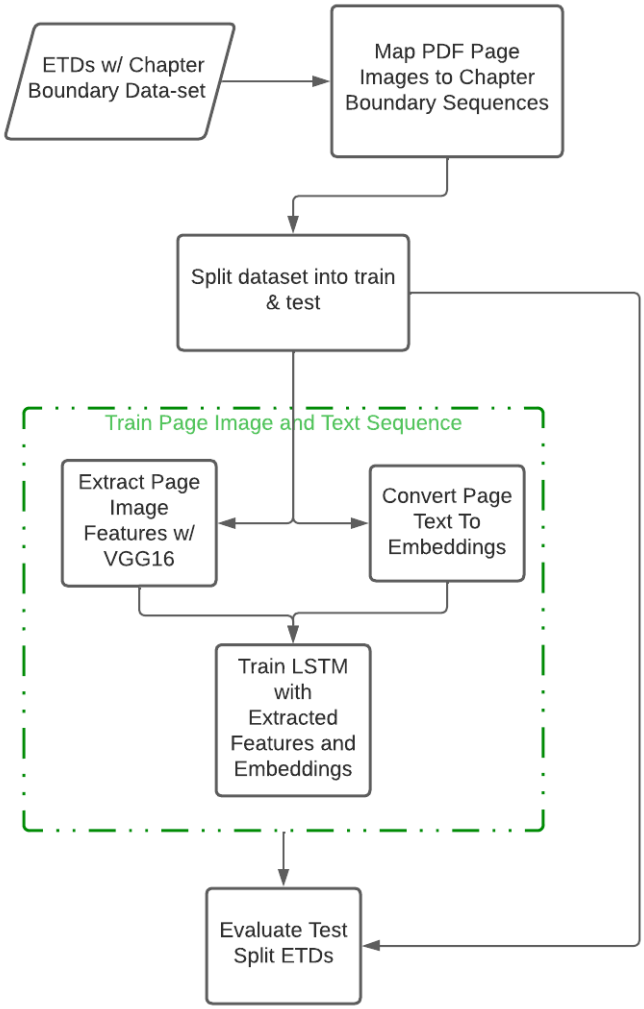


Figure 6.7: Image and Text sequences: Data-flow diagram

in-element page. *Scikit-learn's* metrics library was used to collect and compare model performance [30].

6.6 Training, Validation, and Testing Splits

Each model is trained, validated, and tested using the same split. The training and validation is done upon the overall dataset (1,459 ETDs) by dividing it according to an 80/20 split.

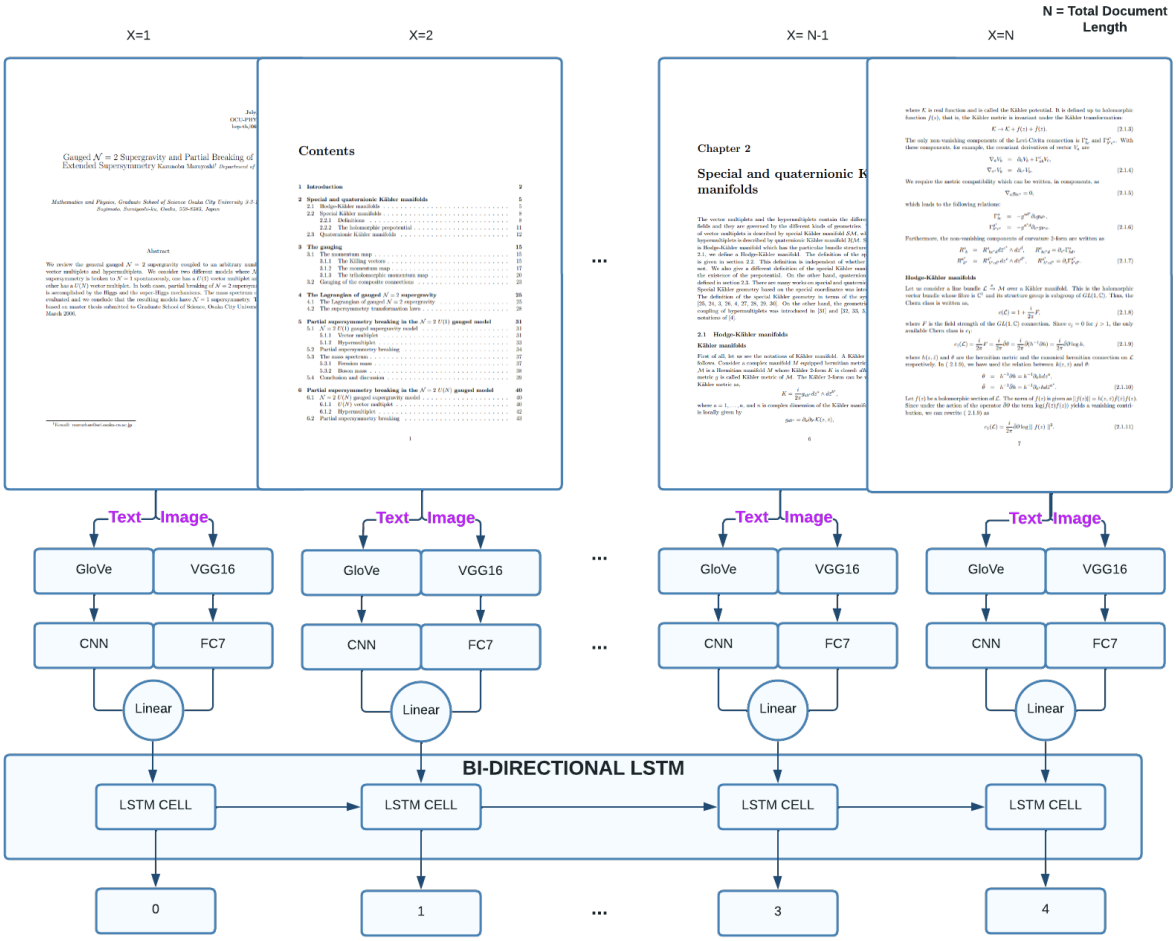


Figure 6.8: Image and Text sequence tagger: Model architecture

The training set is used to train the model weights and biases, and the validation set is used to evaluate training performance for each epoch. After training is over (15 epochs) the models are evaluated on the manually-labelled dataset (150 ETDs).

Chapter 7

Experiments, Results, and Discussion

In this chapter, we discuss various experiments we conducted. We begin our discussion by comparing the performance of other tools that can be used for segmenting ETDs into chapters. These tools include GROBID [22]. Following this, we discuss the performance of our deep learning models. As discussed in our previous chapters, we have designed two slightly different models. The first model extracts features from image sequences as inputs, while the latter (more robust) model uses extracted image features and page text embeddings as inputs.

The following experiments were performed on *Google Colab* [6] using their **Pro+ subscription**. The Pro+ subscription allows users to access GPUs more readily and to employ longer runtimes (24 hours). It also allows for background executions, which was necessary as training the following models took days. Pro+ allows a High-RAM environment of 52 GB to users. The GPU allocation was randomly assigned based on availability. We would randomly be assigned one of the following GPUs: K80, T4, or P100.

7.1 Comparison with Other Segmentation Tools

This section discusses and compares the performance of the Chapter-Segmentation Pipeline with other available segmentation tools including GROBID. The comparison is conducted on a set of 50 randomly sampled ETDs that the CSP successfully generated chapter bound-

aries for. Additionally, chapter boundaries for this set of ETDs were manually labelled and can be considered as the ground truth. This allowed us to make comparisons with other segmentation tools.

7.1.1 GROBID

GROBID is a machine learning library that can be used to extract, parse, and even restructure documents [22]. Its primary use has been with technical and scholarly documents. Our focus is on theses and dissertations. It can be used to used to extract much important structural information from PDFs including segmentation details. It outputs details in XML/TEI encoded documents.

GROBID's setup is much simpler then our Chapter-Segmentation Pipeline and its only required input is the PDF of the document itself. The output XML/TEI file is a structured document containing segmentation, text within each segments, and much more.

```
<div
    xmlns="http://www.tei-c.org/ns/1.0">
    <head>Chapter 1 Introduction</head>
    <p>Physical phenomena occurring in high energy physics are analysed in terms of 'particles', ari
        <ref target="#b56" type="bibr">[57]</ref>. He gives a complete classification of all these r
        <ref target="#b52" type="bibr">[53]</ref>, how mass and spin of a particle are to be describ
        <ref target="#b28" type="bibr">[29,</ref>
        <ref target="#b12" type="bibr">13]</ref>. An outline of the underlying mechanism, following
        <ref target="#b12" type="bibr">[13]</ref>, may be appropriate at this point. Due to Gauss' l
        <ref target="#b16" type="bibr">[17]</ref>, it is then possible to classify the pure particle
    </p>
</div>
<div
    xmlns="http://www.tei-c.org/ns/1.0">
    <head>Assumptions of Local Quantum Physics</head>
    <p>We collect here the main structural postulates upon which Local Quantum Physics is built in t
        <ref target="#b32" type="bibr">[33,</ref>
        <ref target="#b1" type="bibr">2]</ref>, principally in order to fix notation.
    </p>
</div>
<div
```

Figure 7.1: GROBID sample XML/TEI file

Figure 7.1 shows a snippet of a TEI file GROBID produced. The hierarchical structure contains segmentation titles in the header tag. The header tag can also contain enumeration details if the PDF is formatted to include it. Unlike the CSP, GROBID can also provide results on sections within chapters, providing further granularity. The results of GROBID could theoretically be used similar to the CSP and mapped back onto the original PDF. However, the results of GROBID contained some issues that could make it unreliable to confidently map back to the original PDF. The list below describes some of these errors:

- Includes heading, but that is ambiguous regarding if chapter, section, or subsection.
- Includes figure heading in segmentation detail.
- Duplicate entries.
- Includes equations as headings.
- Includes random text as heading.
- No distinction between front-matter, chapters, and end-matter.

Table 7.1 displays generation metrics for GROBID segmentation of the set of 50 manually labelled ETDs. This set of ETDs was extracted from the set of documents (1,459 ETDs) that the CSP successfully worked upon.

	Number of Documents
No TEI File Produced	15
Empty TEI file	27
Successfully Generated Document Details	8

Table 7.1: GROBID: TEI file metrics

Figure 7.2 shows a segmentation comparison for the 8 ETDs that successfully generated segmentation details. The green bar shows the number of chapters manually labelled for

each ETD. GROBID documentation does not claim chapter segmentation. The results make clear that GROBID does not provide a good solution for segmenting ETDs into chapters.

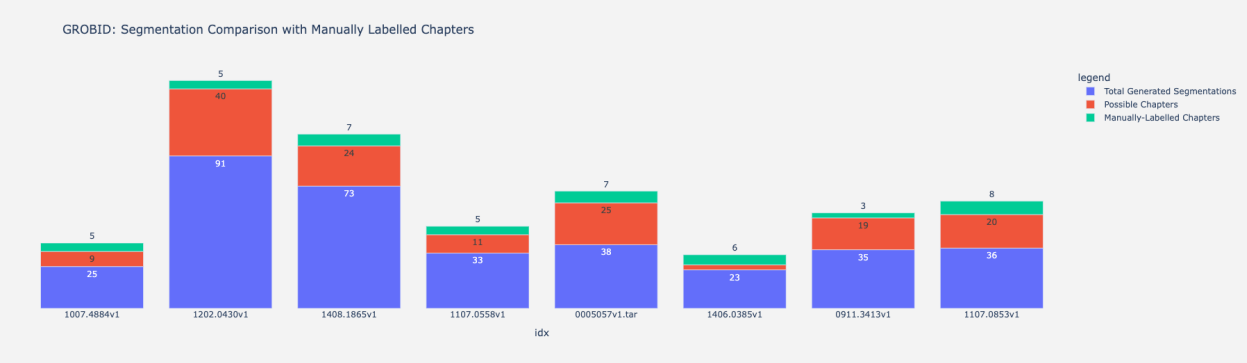


Figure 7.2: GROBID: Segmentation results for successful documents

7.2 Experiment 1: Image-only Sequences

7.2.1 Training Setup

Our first deep learning model is an image sequence tagger. Given a PDF of an Electronic Thesis or Dissertation, pages were converted into a sequence of PNG images similar to sequences of frames in a video. This was done using a Python package called **pdf2image** [5], and the image resolution was set to 100 DPI. The label for each page image was mapped accordingly using our generated chapter boundary details. Each page could be distinctly labelled between six possible outcomes. These include FM element start, FM normal page, chapter start, chapter normal page, EM element start, and EM page. For a detailed visual of our Image-only model architecture refer to Figure 6.4.

The Chapter-Segmented ETD data set was divided in a 80:20 training and validation split (see Section 6.6 for more details). The model was initiated using the parameters listed in

Table 7.2. We performed testing and evaluation upon our manually labelled data set that we discuss in Section 5.3.

Parameter	Values
Embedding Dimensions	4096
Hidden Dimensions	2048
Output Dimensions	6
Dropout	0.25
Bidirectional	True

Table 7.2: Experiment 1: Model parameters

7.2.2 Training

Given the relatively long nature of ETDs and the hardware limitations we had while training our models, we had to divide our training input sequences into mini-batches of 10 pages at a time. Images were transformed using **PyTorch's** transformation toolkit as mentioned in Section 6.2.1. The model was trained over 15 epochs. Each epoch took roughly 3.5 hours to complete, so it took a little over two days to complete training for 15 epochs.

Figure 7.3 displays the training loss over each epoch. From the graph, it's evident that our first model of only image sequences was not learning well. The training loss quickly converges and does not show a consistent decrease of loss over epochs.

In other experiments, we retrained our image-sequence tagger from scratch and altered various variables but witnessed similar results. As a result of no decrease in overall model loss during training, our model's F1 scores and other metrics showed poor performance when applying our model onto our manually labelled data set. Our model could not differentiate between page images of different classes. The low training performance for our Image-only

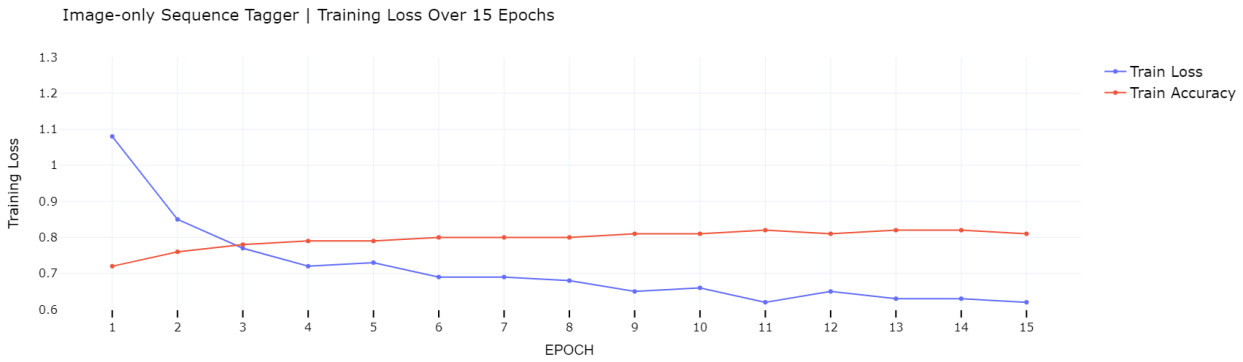


Figure 7.3: Experiment 1: Training loss

sequence tagger were disappointing. In Section 7.4 we discuss an image and text sequence tagger that shows significantly better results.

7.3 Experiment 2: Text-only Sequences

The next deep learning model is a page-text sequence tagger. Given a PDF of an Electronic Thesis or Dissertation, the text of each page was embedded as a sequence. The text was extracted using *PyMuPDF* [4] and the embeddings had 100 dimensions. For a detailed visual of the Text-only tagger's model design refer to Figure 6.6.

7.3.1 Training Setup

Parameter	Values
Text Embedding Dimensions	100
Hidden Dimensions	2048
Output Dimensions	6
Dropout	0.25
Bidirectional	True

Table 7.3: Experiment 2: Model parameters

Similarly to the first model, this model was trained on *Google Colab* [6] servers using GPUs. The input was single batched ETDs and the pages were further batched by 10 pages. The servers could not handle the input of an entire ETD except through batches. The total time to train an epoch was nearly an hour. The pre-trained weights of the vocabulary were kept frozen until the sixth epoch to allow the other model parameters to settle [8].

Training using Text-only sequences show poor results with almost no reduction in training loss. In fact, by the end of training the model showed to increase in loss slightly. During evaluation the model would classify everything as chapter contents, which is also the majority class. The graph in Figure 7.4 shows the training loss over 15 epochs.

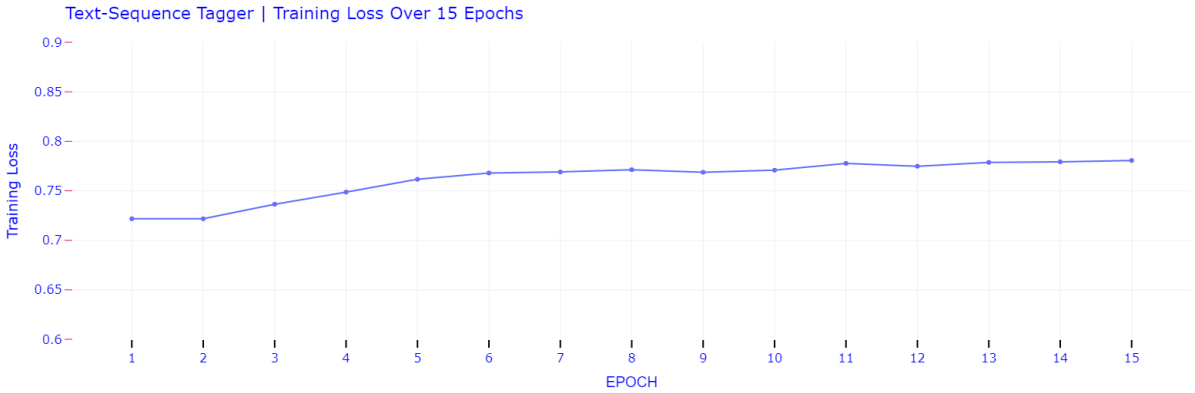


Figure 7.4: Experiment 2: Training loss

7.4 Experiment 3: Image and Text Sequences

7.4.1 Training Setup

Next, we adjusted our models to also include image and text for pages. Page images were pre-processed, transformed, and prepared in the same manner as described in Experiment 7.2. Additionally, each page's text was extracted using a Python package called PyMuPDF

[4]. Page text was then cleaned-up, tokenized, and converted using each of two different embeddings. We discuss our text preparation process in Section 6.2.2.

Similar to Experiment 7.2.1, we randomly assigned our data set into a 80-20 training and validation split. We evaluated our model's performance using our manually labelled data set. For a design overview of our page image and text sequence tagger, refer to Figure 6.8.

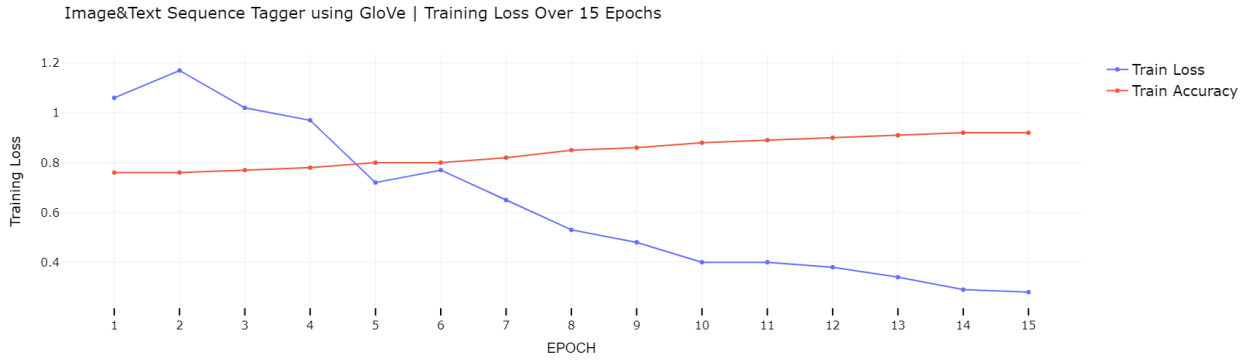
7.4.2 Training

As mentioned regarding our previous experiment, ETDs are generally lengthy documents and so we had to divide our input sequences into mini-batches as available GPUs could not handle the load. Image features were extracted using pre-trained VGG16 and text was prepared and converted into two different embeddings, Glove and fastText, in two different experiments. The preparation process, model dimensions, and training for both experiment cases were identical, to provide comparison between the two embeddings. Each model was trained over 15 epochs. For both embeddings, total training time was similar. It took roughly 5 days to complete training as a single epoch took around 5.5 hours to complete. The model was instantiated using the parameters mentioned in Table 7.4.

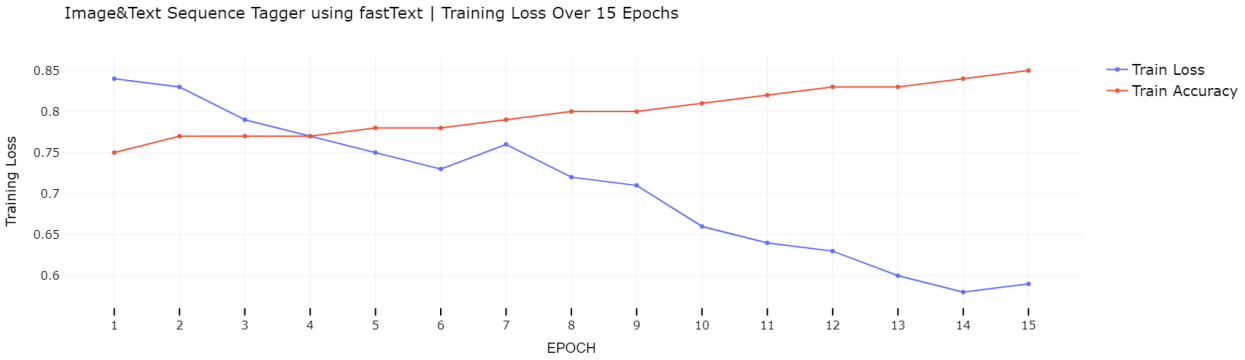
Parameter	Value
Extracted Image Feature Dimensions	4096
Converted Text Embedding Dimensions	100
Hidden Dimensions	2048
Number of Layers	2
Bidirectional	True

Table 7.4: Experiment 3: Model parameters

Figure 7.5 show the training loss and accuracy of the two separate experiments over 15 epochs. Adding textual context significantly increased the training rate for the models.



(a)

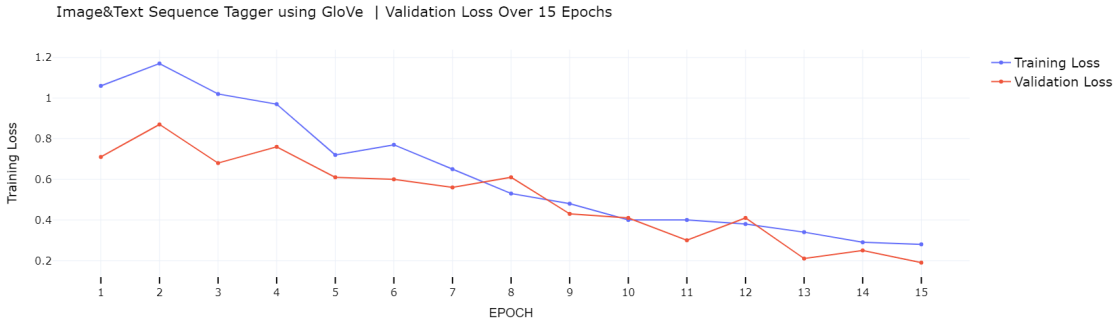


(b)

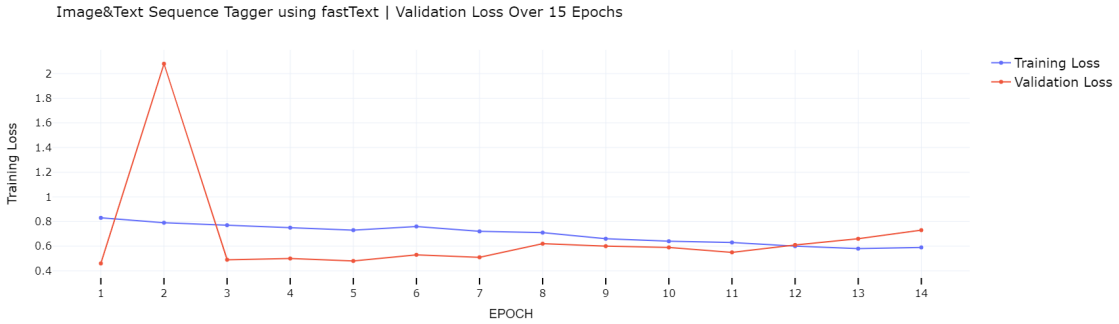
Figure 7.5: Training loss vs. Accuracy using (a) GloVe (b) fastText

Allowing our pre-trained embedding parameters to train, led to no abrupt changes in training. The effects of unfreezing parameters to fine-tune the embeddings can be seen starting in epoch 6 for both experiments.

Additionally Figure 7.6 displays validation scores in comparison over the same epochs. GloVe validation loss is consistently below the training loss, while fastText shows spikes and abrupt changes. In the next section, we will compare the performance of each embedding using the manually labelled data set discussed in Section 5.3.



(a)



(b)

Figure 7.6: Training loss vs. Validation loss (a) GloVe (b) fastText

7.4.3 Results

This section discusses testing results of applying the models trained above, using the manually labelled data set. It also provides further comparison on testing metrics between documents from arXiv and outside sources. As mentioned before, there are in total 6 possible classes. For each class the tables below will show the rate of true positives (TP), false positives (FP), false negatives (FN), precision, recall, and F1 scores. Details on each of these metrics can be found in Section 6.5.

Models on manually labelled data set

Table 7.5 shows results from running the model using GloVe embeddings onto the manually labelled data set. The results show higher precision values, while the recall rate is lacking (especially for front-matter elements). Unbalanced data sets can lead to this. This can also be attributed to the Chapter-Segmentation Pipeline's inefficiency with capturing front-matter elements, as discussed in Section 5.1.

Page Type	TP	FP	FN	Precision	Recall	F1
FM Chapter-Start	171	107	344	62%	33%	**43%**
FM PAGE	480	412	727	54%	40%	**46%**
Chapter-Start	612	199	345	75%	64%	**69%**
Chapter PAGE	15724	2357	596	87%	96%	**91%**
EM Chapter-Start	211	129	205	62%	51%	**56%**
EM PAGE	1309	368	1355	78%	49%	**60%**

Table 7.5: Model performance with GloVe embeddings on manually labelled data set

Table 7.6 also shows metrics by testing the performance of the model using fastText embeddings. fastText in comparison has overall higher precision over GloVe.

Page Type	TP	FP	FN	Precision	Recall	F1
FM Chapter-Start	149	53	366	74%	29%	**42%**
FM PAGE	583	551	624	51%	48%	**50%**
Chapter-Start	465	60	492	89%	49%	**63%**
Chapter PAGE	15802	2599	518	86%	97%	**91%**
EM Chapter-Start	192	23	224	89%	46%	**61%**
EM PAGE	1220	382	1444	76%	46%	**57%**

Table 7.6: Model performance with fastText embeddings on manually labelled data set

Performance Comparison

As mentioned in Section 5.3, the manually labelled data set contains 100 ETDs that were collected from arXiv, and 50 ETDs that were collected from outside sources. Here we provide a performance comparison for our model on arXiv documents and non-arXiv documents. Refer to Tables 7.7 and 7.8 for comparisons.

GloVe Performance Comparison						
Page Type	arXiv			non-arXiv		
	Precision	Recall	**F1**	Precision	Recall	**F1**
FM Chapter-Start	64%	47%	**54%**	56%	20%	**29%**
FM PAGE	65%	44%	**52%**	38%	32%	**35%**
Chapter-Start	82%	77%	**79%**	58%	38%	**46%**
Chapter PAGE	91%	97%	**94%**	80%	94%	**86%**
EM Chapter-Start	78%	68%	**73%**	23%	17%	**19%**
EM PAGE	83%	64%	**72%**	67%	31%	**42%**

Table 7.7: GloVe embeddings: arXiv vs. non-arXiv ETDs

fastText Performance Comparison						
Page Type	arXiv			non-arXiv		
	Precision	Recall	**F1**	Precision	Recall	**F1**
FM Chapter-Start	83%	49%	**62%**	46%	0.09%	**15%**
FM PAGE	66%	51%	**57%**	35%	44%	**39%**
Chapter-Start	97%	64%	**77%**	53%	17%	**26%**
Chapter PAGE	89%	99%	**94%**	80%	93%	**86%**
EM Chapter-Start	95%	64%	**76%**	54%	11%	**18%**
EM PAGE	85%	55%	**67%**	63%	35%	**45%**

Table 7.8: fastText embeddings: arXiv vs. non-arXiv ETDs

arXiv contains a selected set of disciplines that are discussed in Section 3.1. Since the models are trained using arXiv documents, the model performs significantly better on documents within arXiv for each embedding. GloVe slightly outperforms fastText at identifying

chapters, while fastText shows an upper hand at identifying front-matter and end-matter element-starts.

Chapter 8

Conclusion

This research focuses on segmenting chapters within Electronic Theses and Dissertations (ETDs). We begin by describing our research problem and its motivation. We formulate research questions and review various works, tools, and literature relevant to this work.

There has been little research focus on the structure of long scholarly documents. There are currently no publicly available data sets of ETDs with their chapter boundaries. To address this, we explored modifying LaTeX source files to help generate chapter boundary details. Our initial attempt failed to produce a pipeline that could provide consistent results at scale. Our latest Chapter-Segmentation Pipeline works well, using ground truth context to verify selections. This process also distinguishes among front-matter, end-matter, and chapters. We provide extensive details on manually inspecting the results of the CSP (see Section 5.1 for details).

We provide access to a repository that contains 1,459 ETDs, with chapter boundaries for each document. Additionally, we provide access to a data set containing 150 ETDs with manually labelled chapter boundaries.

We then use the first newly created data set to train various deep learning models to predict chapter boundaries of a given ETD. We discuss how we pre-processed each page's image and text to prepare input sequences for each ETD. We next test the image sequence model, the text-based model, and the combination image and text sequence model.

Finally, we discuss the results of our experiments testing those models on our second data set. We evaluated our models using precision, recall, and F1 scores. We observed that combining image and text inputs greatly increases the chance of correct classification using deep learning models.

In the next chapter, we discuss possible future works to expand research in this direction. We provide motivation for creating meaningful repositories of scholarly documents that include annotation details on various elements.

Chapter 9

Future Work

This research was conducted on a fairly small sized corpus of Electronic Theses and Dissertations to begin with, i.e., **2,838** ETDs. This was thanks to the hard work by Virginia Tech's Ph.D. candidate Bipasha Banarjee. She had manually collected and verified these ETDs that were taken from arXiv. Since source LaTeX files are a requirement of our Chapter-Segmentation Pipeline (CSP), there are few repositories that provide these files. arXiv also does not contain any public metadata information to reliably qualify published documents as ETDs. Using the CSP, we were able to generate segmentation details on 1,459 ETDs. Given a bigger set of ETDs with source files, we could have generated a bigger set of chapter segmented ETDs. A larger set of chapter segmented ETDs from various disciplines could be used to train deep learning models to be more general at the task of chapter segmentation. The CSP also has the potential to generate chapter boundary details of other types of documents. It could be used to create a general repository of scholarly documents segmented into chapters.

As discussed in Section 7.2.1, training our deep learning models, even with a small set of ETDs, was fairly expensive, requiring GPUs, and we were not able to tune our hyper-parameters. Since the task of chapter segmentation requires a large sequence of pages with various input elements, tuning the hyper-parameters could greatly improve the performance of our models. There could also be alternate deep learning designs that could be implemented

that could improve the performance of models. The designs of the single input models for image or text sequences need to be improved too, as they showed poor training results.

Modifying LaTeX has been used prior to our work to help output important positional information for document elements such as figures and tables [16, 26]. We have shown that LaTeX modifications can also help output chapter boundary information. Further efforts in this direction could lead to the annotation of various other key elements such as equations, titles, and code markings. It would be interesting to see the creation of a Gold Standard data set that contain more granular information than chapter boundaries, such as locations of title tags, figure captions, and equations. To make this possible and to increase representation, it would be helpful for schools and universities to encourage/require any submitter to include source files for each document. Additionally, the Chapter-Segmented ETD data set could be augmented by further modifying the source LaTeX files. This could be used to train the deep learning models on a larger set of ETDs.

Finally, improvements could be made in our segmentation pipeline. We worked on two separate pipelines to generate segmentation details. There are most likely many other ways to explore LaTeX manipulations to generate chapter boundary details. While our Delimiter-based strategy for segmentation provides accurate chapter boundary details, it could only work on about half of the documents in our repository. A strong background in LaTeX programming possibly could help provide a more robust code that works well across various LaTeX classes.

Bibliography

[1] Md Zahangir Alom, Tarek M. Taha, Chris Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Mahmudul Hasan, Brian C. Van Essen, Abdul A. S. Awwal, and Vijayan K. Asari. A state-of-the-art survey on deep learning theory and architectures. *Electronics*, 8(3), 2019. ISSN 2079-9292. doi: 10.3390/electronics8030292. URL https://www.mdpi.com/2079-9292/8/3/292. Last accessed on 10/15/2022.

[2] Isaac Alpizar-Chacon and Sergey Sosnovsky. Expanding the Web of Knowledge: One Textbook at a Time. In *Proceedings of the 30th on Hypertext and Social Media*, HT '19, New York, NY, USA, 2019. ACM. https://www.db.ics.keio.ac.jp/seminar/2020/20200519_shinzato/Expanding%20the%20Web%20of%20Knowledge.pdf, last accessed on 07/20/2022.

[3] Isaac Alpizar-Chacon and Sergey Sosnovsky. Order out of Chaos: Construction of Knowledge Models from PDF Textbooks. In *Proceedings of the ACM Symposium on Document Engineering 2020*, 2020. https://dl.acm.org/doi/10.1145/3395027.3419585, last accessed on 07/20/2022.

[4] Artifex. PyMuPDF, 2022. https://pymupdf.readthedocs.io/en/latest/, last accessed on 07/05/2022.

[5] Edouard Belval. pdf2image, 2021. https://pypi.org/project/pdf2image/, last accessed on 03/05/2022.

[6] Ekaba Bisong. *Google Colaboratory*. Apress, Berkeley, CA, 2019. ISBN 978-1-4842-4470-8. doi: 10.1007/978-1-4842-4470-8_7. URL https://doi.org/10.1007/978-1-4842-4470-8_7. Last accessed on 04/23/2022.

[7] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomás Mikolov. Enriching word vectors with subword information. *arXiv:1607.04606 [cs.CL]*, 2016. doi: 10.48550/arXiv.1607.04606. URL http://arxiv.org/abs/1607.04606. Last accessed on 04/20/2022.

[8] Prashant Brahmbhatt. Using Pre-trained Models Effectively, 2020. https://codingshogun.com/using-pre-trained-models-effectively/, last accessed on 07/20/2022.

[9] TEI Consortium. Guidelines for electronic text encoding and interchange, Apr 2017. URL https://www.tei-c.org/release/doc/tei-p5-doc/en/html/index.html. Last accessed on 02/15/2022.

[10] Continuum analytics. Anaconda software distribution, 2020. URL https://docs.anaconda.com/. Last accessed on 05/15/2022.

[11] Arden Dertat. Applied Deep Learning - Part 1: Artificial Neural Networks, Oct 2017. https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6, last accessed on 07/28/2022.

[12] David Evans. PostScript vs. PDF, 2022. https://www.asc.ohio-state.edu/schumacher.60/imageinfo/pdf_ps_eps.html, last accessed on 07/26/2022.

[13] Mathieu Fenniak. Welcome to PyPDF2, 2022. https://pypi.org/project/PyPDF2/, last accessed on 07/05/2022.

[14] John Hammersley. Overleaf, Online LaTeX Editor, 2022. https://www.overleaf.com/, last accessed on 09/10/2022.

[15] Palukh Jude. Increasing Accessibility of Electronic Theses and Dissertations (ETDs) Through Chapter-level Classification. *Virginia Tech Department of Computer Science*

M.S. thesis, Blacksburg, VA 24061 USA, 07 2020. http://hdl.handle.net/10919/99294, last accessed on 10/10/2020.

[16] Sampanna Yashwant Kahu. Figure extraction from scanned electronic theses and dissertations. *Virginia Tech Department of Computer Science M.S. thesis, Blacksburg, VA 24061 USA*, Sep 2020. http://hdl.handle.net/10919/99294, last accessed on 05/06/21.

[17] Jonathan Kew. XeTeX - TeX Users Group, 2022. https://tug.org/xetex/, last accessed on 03/15/2022.

[18] Asifullah Khan, Anabia Sohail, Umme Zahoora, and Aqsa Saeed Qureshi. A survey of the recent architectures of Deep Convolutional Neural Networks, Apr 2020. https://link.springer.com/article/10.1007/s10462-020-09825-6, last accessed on 10/15/2021.

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf. Last accessed on 09/20/2022.

[20] Leslie Lamport. LaTeX: A document preparation system, 2022. https://www.LaTeX-project.org/, last accessed on 06/15/2022.

[21] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. May 2015. doi: 10.1038/nature14539. https://doi.org/10.1038/nature14539, last accessed on 09/19/2022.

[22] Patrice Lopez. Grobid. https://github.com/kermitt2/grobid, 2022. https://grobid.readthedocs.io/en/latest/, last accessed on 07/15/2022.

[23] Chris Mattman. Tika-Python, 2022. https://github.com/chrismattmann/tika-python, last accessed on 07/03/2022.

[24] Seyyid Ahmed Medjahed. A comparative study of feature extraction methods in images classification. 2015. doi: 10.5815/ijigsp.2015.03.03. https://www.mecs-press.org/ijigsp/ijigsp-v7-n3/v7n3-3.html, last accessed on 09/10/2022.

[25] James A Nichols, Hsien W Herbert Chan, and Matthew A B Baker. Machine learning: Applications of artificial intelligence to imaging and diagnosis, Feb 2019. https://link.springer.com/article/10.1007/s12551-018-0449-9, last accessed on 09/10/2022.

[26] Russell Power Waleed Ammar Noah Siegel, Nicholas Lourie. Extracting scientific figures with distantly supervised neural networks: Proceedings of the 18th ACM/IEEE joint conference on digital libraries, May 2018. https://dl.acm.org/doi/abs/10.1145/3197026.3197040, last accessed on 08/14/2021.

[27] Razvan Pascanu, Tomas Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. *arXiv:1211.5063 [cs.LG]*, 2012. doi: 10.48550/arXiv.1211.5063. URL https://arxiv.org/abs/1211.5063. Last accessed on 04/04/2022.

[28] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In

Advances in Neural Information Processing Systems 32, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf. Last accessed on 06/05/2022.

[29] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. *arXiv:1705.04304 [cs.CL]*, 2017. doi: 10.48550/arXiv.1705.04304. URL http://arxiv.org/abs/1705.04304. Last accessed on 04/10/2022.

[30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. Last accessed on 08/20/2021.

[31] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014. https://nlp.stanford.edu/projects/glove/, last accessed on 07/28/2022.

[32] Sebastian Rahtz and Karl Berry. TeX Live, 2022. https://www.tug.org/texlive/, last accessed on 04/15/2022.

[33] Greg Roelofs. PNG (Portable Network Graphics) Home Site. http://www.libpng.org/pub/png/, last accessed on 07/25/2022.

[34] Iqbal H Sarker. Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions. *SN computer science*, 2021. doi: 10.1007/s42979-021-00815-1. 2, 420 https://link.springer.com/article/10.1007/s42979-021-00815-1, last accessed on 09/10/22.

[35] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556 [cs.CV]*, 2014. doi: 10.48550/arXiv.1409.1556. URL https://arxiv.org/abs/1409.1556. Last accessed on 03/15/2022.

[36] Iuliana Tabian, Hailing Fu, and Zahra Sharif Khodaei. A convolutional neural network for impact detection and characterization of complex composite structures. *Sensors*, 19 (22), 2019. ISSN 1424-8220. doi: 10.3390/s19224933. URL https://www.mdpi.com/1424-8220/19/22/4933. Last accessed on 09/18/2022.

[37] Hàn Thế Thành. pdfLaTeX, 2022. https://tug.org/applications/pdftex/, last accessed on 04/20/2022.

[38] Ben Trevett and Antonio Sejas. Pytorch pos tagging. https://github.com/bentrevett/pytorch-pos-tagging, 2021. Last accessed on 11/10/2022.

[39] Cornell University. arXiv.org e-Print archive, 2022. https://arxiv.org, last accessed on 04/15/2022.

[40] Cornell University. arXiv Dataset - Kaggle, June 2022. https://www.kaggle.com/dataset/1b6883fb66c5e7f67c697c2547022cc04c9ee98c3742f9a4d6c671b4f4eda591, last accessed on 07/15/2022.

[41] Guido Van Rossum and Fred L Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995. https://docs.python.org/3/reference/, last accessed on 07/15/2022.

[42] Lulu Wan, George Papageorgiou, Michael Seddon, and Mirko Bernardoni. Long-length legal document classification. *arXiv:1912.06905 [cs.CL]*, 2019. doi: 10.48550/arXiv.1912.06905. URL http://arxiv.org/abs/1912.06905. Last accessed on 04/14/2022.

[43] Congcong Wang, Paul Nulty, and David Lillis. A comparative study on word embeddings in deep learning for text classification. In *Proceedings of the 4th International Conference on Natural Language Processing and Information Retrieval*, NLPIR 2020, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450377607. doi: 10.1145/3443279.3443304. URL https://doi.org/10.1145/3443279.3443304. Last accessed on 09/12/2022.