Improving Access to ETD Elements Through Chapter Categorization and Summarization

Bipasha Banerjee

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

Doctor of Philosophy
in
Computer Science and Applications

Edward A. Fox, Chair
Jian Wu
Ismini Lourentzou
Debswapna Bhattacharya
Dawei Zhou

July 11, 2024
Blacksburg, Virginia 24061

Keywords: Summarization, Classification, Natural Language Processing, Machine Learning, Language Models

Improving Access to ETD Elements Through Chapter Categorization and Summarization

Bipasha Banerjee

(ABSTRACT)

The field of natural language processing and information retrieval has made remarkable progress since the 1980s. However, most of the theoretical investigation and applied experimentation is focused on short documents like web pages, journal articles, or papers in conference proceedings. Electronic Theses and Dissertations (ETDs) contain a wealth of information. These book-length documents describe research conducted in a variety of academic disciplines. While current digital library systems can be directly used to find a document of interest, they do not also facilitate discovering what specific parts or segments are of particular interest. This research aims to improve access to ETD components by providing users with chapter-level classification labels and summaries to help easily find portions of interest. We explore the challenges such documents pose, especially when dealing with a highly specialized academic vocabulary. We use large language models (LLMs) and fine-tune pre-trained models for these downstream tasks. We also develop a method to connect the ETD discipline and the department information to an ETD-centric classification system. To help guide the summarization model to create better chapter summaries, for each chapter, we try to identify relevant sentences of the document abstract, plus the titles of cited references from the bibliography. We leverage human feedback that helps us evaluate models qualitatively on top of using traditional metrics. We provide users with chapter classification labels and summaries to improve access to ETD chapters. We generate the top three classification labels for each chapter that reflect the interdisciplinarity of the work in

ETDs. Our evaluation proves that our ensemble methods yield summaries that are preferred by users. Our summaries also perform better than summaries generated by using a single method when evaluated on several metrics using an LLM-based evaluation methodology.

Improving Access to ETD Elements Through Chapter Categorization and Summarization

Bipasha Banerjee

(GENERAL AUDIENCE ABSTRACT)

Natural language processing (NLP) is a field in computer science that focuses on creating artificially intelligent models capable of processing text and audio similarly to humans. We make use of various NLP techniques, ranging from machine learning and language models, to provide users with a much more granular level of information stored in Electronic Theses and Dissertations (ETDs). ETDs are documents submitted by students conducting research at the culmination of their degree. Such documents comprise research work in various academic disciplines and thus contain a wealth of information. This work aims to make such information stored in chapters of ETDs more accessible to readers through the addition of chapter-level classification labels and summaries. We provide users with chapter classification labels and summaries to improve access to ETD chapters. We generate the top three classification labels for each chapter that reflect the interdisciplinarity of the work in ETDs. Alongside human evaluation of automatically generated summaries, we use an LLM-based approach that aims to score summaries on several metrics. Our evaluation proves that our methods yield summaries that users prefer to summaries generated by using a single method.

Dedication

This is dedicated to my parents, Dr. Subir Kumar Banerjee and Mrs. Banani Banerjee. Thank you for always being supportive and teaching me to be resilient. Ma, Baba, I am everything because of you and nothing without you.

Acknowledgments

I would like to express my gratitude to my advisor Dr. Edward A. Fox. Thank you, Dr. Fox, for your continued support and encouragement. Your valuable feedback and insights throughout my degree have been exceptionally crucial in shaping this work. I would also like to thank my committee members, Dr. Lourentzou, Dr. Zhou, Dr. Bhattacharya, and Dr. Wu, for their valuable suggestions and guidance throughout the whole process. A special thanks go to William A. Ingram for his constant support and suggestions throughout the research project. I would also like to thank Virginia Tech University Libraries for supporting me throughout my graduate degree. I want to acknowledge the Department of Computer Science for the opportunity to pursue my doctoral degree. I am very grateful to Sharon for always helping with my questions and calming me down in many situations.

A special thanks to Palakh and Maanav for being my biggest cheerleaders. I would also like to thank Frank for being my confidant and friend during my Ph.D. years. I am grateful to my best friend Sulagna for always being my sounding board and sanity check. I would like to thank Satvik, Prashant, Ola, Xinyue, and other Digital Library Research Laboratory colleagues for their support and suggestions throughout the years.

Lastly, thank you to my family, which has been a constant source of inspiration and support.

This research was made possible in part by the Institute of Museum and Library Services grant LG-37-19-0078-19.

Contents

List of Figures

List of Tables

List of Abbreviations

AI Artificial Intelligence

ETDs Electronic Theses and Dissertations

LLM Large Language Model

LM Language Model

LSTM Long Short-Term Memory

ML Machine Learning

NLP Natural Language Processing

PDF Portable Document Format

PQDT ProQuest Dissertations and Theses

RF Random Forest

SVM Support Vector Machine

Chapter 1

Introduction

1.1 Motivation

Theses and dissertations reflect the work of students and are required at the end of the degree pursued. Starting with Virginia Tech's requirement on January 1, 1997 of electronic theses and dissertations (ETDs), more and more universities require or at least facilitate online submission of these documents. Covering scholarship in all disciplines, ETDs provide opportunities for an in-depth understanding of new discoveries. In terms of detail, an ETD might cover the equivalent of 2-6 journal papers or 5-15 conference papers. Because of their coverage and the cohesion and coherence of expression, they are of particular value to students, faculty, researchers, authors, readers, and others in academia, industry, and government. These book-length documents contain a vast amount of knowledge, are organized into chapters and sections, and are often referred to as 'scholarly long documents' [45]. In this research, we propose to make ETDs more accessible.

Various digital libraries, institutional repositories, and university websites host such documents, often as PDF files along with the metadata. However, such metadata only contains broad document-level information. We introduce services that will make it easier to engage with such long documents and utilize the wealth of research content buried in them.

The domain of information retrieval has made remarkable progress since the 1950s, and

especially since the 1980s. However, most of the theoretical work and experimentation is focused on short documents like web pages, journal articles, or papers in conference proceedings. Search on the resulting systems can be directly used to find a document of interest, but they fail to retrieve the exact part of the document (segment) that contains the result. Therefore, retrieving the entire document is not as useful as retrieving the segment that contributed to the search result. The extensive knowledge inside ETDs is relatively inaccessible to the majority of individuals. For example, while each of the articles in a journal volume or conference proceedings has its own abstract, there are no summaries for the chapters of an ETD. Long documents are challenging to navigate. A conference paper that contains 4-13 pages is easy for an individual to read. On the other hand, reading an entire ETD, which, on average, is at least 100 pages long, takes hours. It is a known fact that almost every ETD starts with an abstract. An abstract might be considered to be a short version of the document. However, a one-page abstract could only act as a very high-level introduction to the topic. We provide summaries and classification labels to the readers at a much more granular level.

1.2 Problem Statement

The work of graduate students reflected in ETDs, is inadequately appreciated, recognized, or cited. This hinders the professional visibility of graduate students and can harm their careers. Since ETDs are not widely used, the reproducibility of scholarship is limited, as few people access the details of graduate research. Due to the limited use of ETDs, university budgets are unduly stretched to cover expensive journals that only contain shorter versions of the work that is covered in more detail in ETDs.

This research aims to make the information buried in ETDs easily discoverable and ac-

cessible. Most digital libraries and university repositories hosting such documents provide document-level metadata information which is in turn used for indexing, searching, and browsing. We want to display more granular metadata information to the user, e.g., at the chapter level. These granular labels and summaries can also be used to obtain more targeted search results. To identify chapters from these long documents, we investigate various segmentation approaches that can effectively detect chapter boundaries. Once segmented, we look into applying summarization and classification to the chapter segments. Thus, a major contribution of this research is to make the data in ETDs more accessible by providing users with chapter-level summaries and classification labels.

1.3 Research Questions

The primary broad research question is: *Can we add value to existing digital library systems by providing more granular level information?* This can be further broken down into the following specific research questions.

1. Can tools that we devise achieve high accuracy in segmenting long documents automatically? This work is discussed in Chapter 3.

2. Can chapter-level classification labels and chapter summaries help in understanding the chapter text better? This work is discussed in Chapters 4 and 5.

3. Can leveraging LLMs and obtaining context from document abstract sentences and cited reference titles from each chapter help in creating better chapter summaries? The work is discussed in Chapters 5 and 6.

4. Can we use human evaluation to create more ground truth data? This work is discussed in Chapter 6.

1.4 Research Hypotheses

The central hypotheses presented in this research are as follows:

1. Our approach to automatic chapter boundary detection using a neural network model trained on long scholarly documents yields better results than available state-of-the-art (SOTA) neural network models trained on collections of shorter documents.

2. Our quantitative experimental results show that a language model built on clean text (i.e., only complete sentences) will result in a model that better understands the scholarly language (as measured by the perplexity score).

3. Our quantitative experimental results show that sophisticated transformer-based language-model classifiers outperform traditional machine learning-based classifiers such as Support Vector Machine and Random Forest.

4. Our quantitative experimental comparison shows that the classification of our ETD chapters using language models improves when fine-tuning SOTA pre-trained models on our corpus instead of directly using SOTA pre-trained models.

5. Our qualitative experimental results show that a multi-label classifier outperforms a multi-class classifier at predicting the correct discipline as measured by accuracy scores.

6. Our experiments regarding prompting and fine-tuning LLMs for classification will generate a better understanding of the chapter content, as shown by our user study.

7. Our experimental results prove that abstractive models with longer context windows generate better summaries than models with shorter context windows, as proved by higher Rouge scores.

8. Our approach of obtaining context from the document abstract and references that correlate with each chapter helps us select a better set of important keywords and sentences and thus create summaries that are scored higher by LLM-based evaluation and are preferred by users.

9. Our approach to using an ensemble summarization technique that incorporates LLMs generates better summaries that are preferred by users to the results of just applying a single abstractive or extractive method.

1.5 Chapter Organization and Overview

The work in this dissertation is organized as follows:

- Chapter 2 summarizes relevant literature related to the research work outlined in this dissertation.

- Chapter 3 talks about the ETD dataset that we have amassed [111] and data subsets that we use in our research. We also discuss several segmentation approaches and our related findings regarding extracting chapters from ETDs. Additional results from Segmentation are discussed in Appendix Section A.

- Chapter 4 proposes our chapter classification framework. We discuss all data subsets used in several experiments to set up baselines and several approaches employed to classify ETD chapters. Supplementary results from experiments in this chapter have been added to the Appendix Section B.

- Chapter 5 proposes several experimental setups for chapter summarization. We also propose a context-aided summarizer in this chapter. This context-aided summarizer

is guided by other document parts, such as selected abstract sentences and cited reference titles. In addition, we propose the use of LLMs to create better ETD chapter summaries. Experimental results are discussed. Auxiliary results from the experiments have been discussed in the Appendix Section C.

- Chapter 6 describes the user study to evaluate chapter labels and chapter summaries of ETDs. We start by discussing two computer science pilot studies and then extend the work to other disciplines across Virginia Tech. Additional materials related to the evaluation setups have been discussed in Appendix Section Dand E.

- In Chapter 7 we discuss in depth our findings in this research, along with future directions.

1.6 Publications and Collaborations

In this section, we list the workshop, conference, and journal publications that resulted in part from this research, indicating for each which of the following chapters are related. We also discuss a variety of collaborations wherein we helped guide efforts that relate to our research, pointing out relationships to the following chapters.

1.6.1 Publication and Talks

1. William A. Ingram, Jian Wu, Sampanna Yashwant Kahu, Javaid Akbar Manzoor, **Bipasha Banerjee**, Aman Ahuja, Muntabir Hasan Choudhury, Lamia Salsabil, Winston Shields, and Edward A. Fox. *"Building datasets to support information extraction and structure parsing from electronic theses and dissertations."* International Journal on Digital Libraries (2024): 1-22 [57]. [Related to Chapter 3]

2. Sung Hee Park, **Bipasha Banerjee**, William A. Ingram, Edward A. Fox. *"Case Study of Analyzing the Variety of ETD Layouts"*. 26th International Symposium ETD 2023, INFLIBNET Centre, Gandhinagar, India, October 26-28, 2023 [89]. [Related to Chapter 3]

3. Satvik Chekuri, Prashant Chandrasekar, **Bipasha Banerjee**, Sung Hee Park, Nila Masrourisaadat, Aman Ahuja, William Ingram and Edward Fox. *''Integrated Digital Library System for Long Documents and their Elements"*. In Proceedings of the 23rd ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL 2023), nominated for Best Student Paper, 13-24, DOI 10.1109/JCDL57899.2023.00012 [27]. [Related to Chapters 4 and 5]

4. **Bipasha Banerjee**, William A. Ingram, Jian Wu, and Edward A. Fox.*"Applications of data analysis on scholarly long documents."* (IEEE Big Data 2022 workshop Computational Archival Science, Osaka, Japan) [16]. [Related to Chapters 4 and 5]

5. **Bipasha Banerjee**, Palakh Mignonne Jude, William A. Ingram, Kurt Luther, Edward A. Fox. *"Help Me Help You - A Mixed-Initiative Approach To Explore Book-length Documents"*. (Talk presented at CIKM 2022 Workshop on Human-in-the-loop Data Curation) [35]. [Related to Chapter 4]

6. **Bipasha Banerjee**. *"Opening scholarly documents through text analytics"*.

   (ACM/IEEE JCDL 2022 Doctoral Consortium) [13]. [Related to Chapters 4, 5, and 6]

7. Sami Uddin, **Bipasha Banerjee**, Jian Wu, William A Ingram, Edward A. Fox. *"Building A Large Collection of Multi-domain Electronic Theses and Dissertations"*. (IEEE Big Data 2021 poster paper) [111]. [Related to Chapter 3]

8. **Bipasha Banerjee**, William A. Ingram, Edward A. Fox. *"Extracting Information from Electronic Theses and Dissertations"*. (Presented at CAPWIC 2021, ACM Capital Region Celebration of Women). [Related to Chapters 3, 4, and 5.]

9. William A. Ingram, **Bipasha Banerjee**, Edward A. Fox. *"Summarizing ETDs with deep learning."* (ETD conference 2019) [56]. [Related to Chapter 5]

1.6.2 Collaborations

There were several successful collaborations with teams in courses taught at Virginia Tech (CS4624 and CS5604), along with researchers at Virginia Tech's Digital Library Research Laboratory Lab (DLRL).

1.6.2.1 CS4624: Multimedia, Hypertext, and Information Access

CS4624 is a capstone undergraduate course taught at Virginia Tech. Students in this course are grouped into teams each assigned to a project, and are assigned to a client (subject matter expert or SME). I have been the client of CS4624 offerings in Spring 2023 and Spring 2024. The work reported in [102] relates to Chapters 3 and 4. The work covered in [47] relates to Chapter 6.

1.6.2.2 CS5604: Information Storage and Retrieval

CS 5604 is a graduate-level course in computer science that follows a problem-based learning structure. As described in the course syllabus, it motivates students to learn "analyzing, indexing, representing, storing, searching, retrieving, processing and presenting information and documents using fully automatic systems". As a subject matter expert for CS5604's

Fall 2022 and 2023 offerings, I mentored students to successfully integrate segmentation, classification, and summarizing services into the working system as reported in [27] and [43]. These courses and related work led to a utility patent application filed early in 2024, in which I am one of the inventors [1]. This work relates to Chapters 3, 4, and 5.

1.6.2.3 DLRL Collaborations

Alongside involvement in CS4624 and CS5604, I was involved in collaborations with other researchers in DLRL. We have investigated several approaches to segmenting ETDs that are discussed in Chapter 3. Work on segmentation using a LaTeX-based method [78] has been reported in our paper [57] and relates to Chapter 3. I have modified and used the research work reported in [5] for segmentation and have extended it into a hybrid text extraction pipeline as discussed in Chapter 3. I have guided and extended the classification work done in [59], which relates to Chapter 4.

Chapter 2

Literature Review

This chapter discusses existing methods, tools, and techniques related to the research discussed in this document.

2.1 Text Extraction Tools

We need to extract the text from our document collection. As mentioned in Chapter 3, our ETD repository collection consists of book-length documents in PDF files and the associated metadata in the form of XML files. Before we can perform text analysis on the documents, we need to extract the text and perform any pre-processing needed for the analysis tasks. Here, we focus our discussion on various tools that can be used for text extraction.

2.1.1 Abbyy Cloud OCR SDK

Abbyy Cloud OCR SDK [2] is a tool to process information from documents and images. It is a web-based service that helps with information extraction via a REST API for documents in over 200 languages. Apart from information extraction, it aids users with processing PDF files, automatic table extraction, and image understanding. The caveat to keep in mind is that the tool is not available to use freely and has to be paid for. Given that we have a set of over 500,000 documents, it would be very expensive to use this tool.

2.1.2 PDFMiner

PDFMiner [42] is a PDF text extraction tool built on Python. PDFMiner attempts to extract text from a PDF file using a combination of a parser and a converter. The parser identifies elements such as pages, fonts, and characters. Once the PDF file has been parsed, the converter is used to extract the text from the identified elements. Though this tool works well for simpler PDF documents, it fails to extract text accurately when the layout is complex such as in texts with images and tables. The output is often inconsistent and thus is unreliable to use for a large corpus of long and complex documents.

2.1.3 PyMuPDF

PyMuPDF [69] is a text extraction tool based on MuPDF [10]. MuPDF is a powerful framework for viewing and converting PDFs, XPS files, and E-books. PyMuPDF is the Python wrapper built on top of MuPDF for PDF text extraction and parsing. The top-level import library for this Python binding is known as 'fitz'. It has the ability to access various file extensions like pdf, xps, oxps, cbs, fb2, or epub. Image documents, that is, documents that have been scanned and are not born digital, are also supported. It allows users to decrypt the document, access meta-information, search for text, render pages, extract text and images, and perform OCR on the text, as well as convert the document to other formats like XML, PDF, JSON, text, and (X)HTML. PyMuPDF, though lightweight, takes up a lot of memory when extracting text from PDFs. We also found it could not process several PDFs in our corpus; it would output empty text files for them. Given the size of our repository and the memory issues, it was not feasible to use this tool at a large scale to extract text from our corpus.

2.1.4 PDFPlumber

PDFPlumber [104] is a Python-based tool to extract text from PDF documents. It has been built on PDFMiner (see above). Although built on PDFMiner, it has added capabilities for extracting detailed information from PDFs that PDFMiner lacks. Information such as table data, text, and character / line information are well extracted with PDFPlumber. It works best for born-digital documents as opposed to scanned documents. The main (object-oriented programming) class of PDFPlumber is the page class. This class enables users not only to extract text from the page but also extract other properties of the page such as page number, width, height, characters, images, etc. The tool allows cropping of text using bounding box information and also extracts additional elements of a PDF file, like tables and words. Our experiments found PDFPlumber to be accurate in extracting text from PDFs.

It is important to note that most Python-based tools work well with born-digital documents but not with scanned documents. Hence, we need to be aware of the problems that might arise with scanned documents. Fortunately, we can use PDFPlumber to extract text from the over 500,000 ETDs in our corpus, which include both born-digital and scanned documents. Further, PDFPlumber excels in its simplicity and ability to utilize multiprocessing. Therefore, scaling the extraction for our large corpus was straightforward and time- and memory-efficient.

2.1.5 AWS Textract

AWS Textract [101] is a text extraction service provided by Amazon Web Services. It uses Amazon's proprietary machine learning algorithm to extract text, handwriting, layout elements, and data from scanned or born-digital PDF files or images. Textract allows users to customize their 'pre-trained or custom' features. Amazon Textract can be used via AWS

SDK, AWS web interface, or AWS API key. Textract outputs line and word information for each page of the PDF. For each of the lines and words, alongside the text, Textract can output a confidence score, bounding box information, and a hierarchy (word to line relation). Textract has been shown to perform well in extracting information from printed documents. Therefore, this makes the most effective, efficient, and accurate way of extracting text from any scanned or born-digital ETD PDF file. The only downside is that the Textract service is not free and needs a subscription.

2.2 Segmentation

We need chapters automatically extracted from our long documents to perform scalable chapter-level analysis on them. Accordingly, we discuss below the obvious approach of trying to leverage tables of contents, and a popular tool used to parse documents. In Chapter 4, we further discuss local work on segmentation in which we have collaborated with two other graduate students.

The 'Table of Contents' (ToC) gives structural information about a document. For ETDs, it contains a detailed listing of chapters, sections, and sub-sections. Thus, the ToC provides a hierarchical representation of the document with several levels of headings included. Several tools like PyMuPDF [69] help extract the ToC contents. ETDs don't have a strict writing format. Therefore, the structure of ToCs also varies greatly. Figure 2.1 shows two such variations in the ToCs. Due to such variations, using a rule-based approach or a tool to extract information from the ToC often results in parsing errors.

Tools such as Grobid [75] help in parsing scientific documents and generating a TEI/XML output. Grobid is a machine learning tool that was trained using scientific papers and articles. That is likely to be why our experiments with ETDs have shown that it is unable to

accurately generate chapter boundaries. As a part of work done in CS 6604 [12], we extracted text from 11674 theses and 7033 dissertations from Virginia Tech's ETD collection[1] using Grobid. We applied post-processing to analyze the TEI/XML results from parsing by Grobid. We found that Grobid generated far more chapters than a document actually had, often recognizing 100-plus chapters from one ETD. We observed a number of chapters extracted being between 1 and 313 with Grobid. Thus, we decided to investigate other approaches for chapter segmentation.

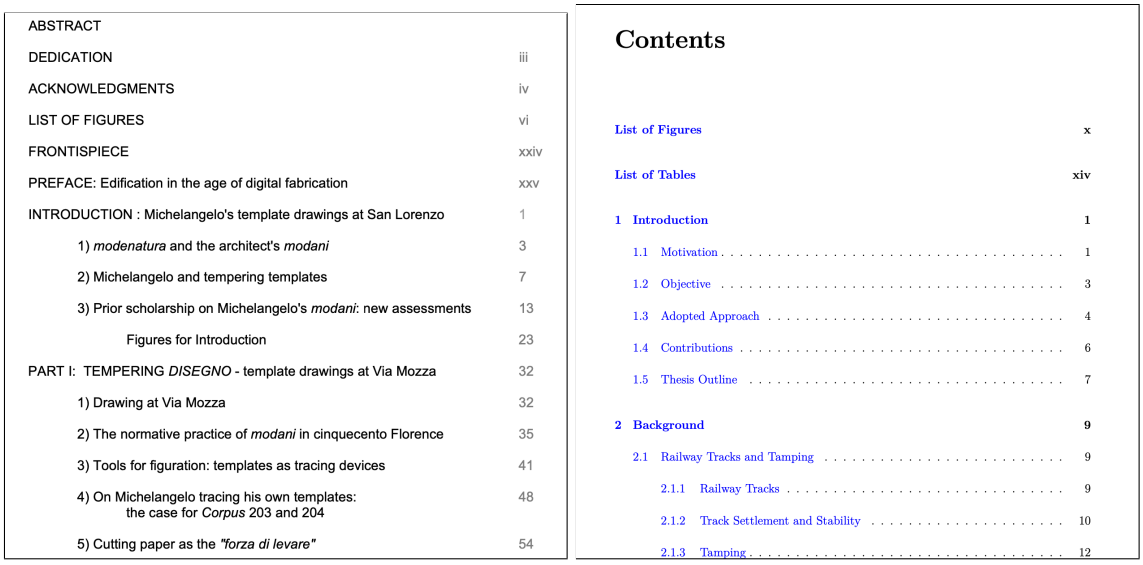


Figure 2.1: Variations in the Table of Contents

2.3 Automatic Classification

In this section, we discuss prior work done in classifying text [99], specifically scholarly documents. Automatic classification is usually a supervised [98] technique of using labeled training data to learn the mapping between features and classes in the output data. Typically, the labeled data is mapped to a pre-defined classification system. We go over some

[1] https://hdl.handle.net/10919/5534

of the classification systems in Section 2.3.1. Classification tasks can be characterized as follows.

1. **Binary Classification** [60] classifies objects into one of two possible classes. A typical example is spam detection which involves classifying an email as either 'spam' or 'not spam.'

2. **Multi-class Classification** [22] is used to classify objects into one of the classes from a known set. An example will be classifying a news article into one of the categories (classes) to which it can belong, e.g., "sports".

3. In **Multi-label Classification** [22], objects can belong to more than one of the classes from the available set of classes. A typical example is a movie belonging to multiple genres (classes). The movie 'SpiderMan' is classified on IMDb [84] as **action**, **adventure**, and **fantasy**.

Popular traditional machine-learning classification techniques include logistic regression, decision trees, naïve Bayes, K-nearest neighbors, support vector machine [19], and random forest [21].

2.3.1 Classification Systems and Classifiers

Several classification systems help to categorize academic disciplines, such as the Library of Congress Classification (LCC) system [79], ProQuest subject categories [91], and ACM Computing Classification System (CCS) [3].

The ACM CCS is a hierarchical taxonomy system for categorizing academic publications in computer science and related disciplines. In his dissertation [28], Yinlin Chen created multiple classifiers to categorize user-generated computing-related resources into the ACM

CCS. The LCC is a hierarchical system used to categorize books and other literature in libraries that is widely used in the United States. The LCC consists of 21 main classes, further divided into sub-classes, etc. Venkat Srinivasan did preliminary studies on classifying ETDs using LCC [107]. ProQuest subject categories provide a hierarchical taxonomy to categorize research material such as theses and dissertations. The major areas include business, health and medical, social sciences, arts, humanities, religion, education, science, and technology. It is further divided into subcategories and specific subjects. In [59], Palakh Jude used various machine and deep learning methods to classify chapters of ETDs into 28 categories using the ProQuest subject categories classification systems. She reported that SVM classifiers performed the best using her document set.

2.4 Language Models and Machine Transduction

In machine transduction [49], the goal is to learn a mapping between the input and output sequences. This involves creating a model that can accurately generate the correct output sequence given any input sequence. Examples of transduction include machine translation, summarization, question answering, and even classification.

We use zero-shot and few-shot learning techniques to perform tasks with limited training data. In zero-shot learning [106, 108], the aim is for an unaided model to correctly classify unseen data. During the training stage, the model learns from auxiliary data in the form of descriptions and other features. On the other hand, in few-shot learning [95, 105], the model is provided with a few examples during the training process. Both methods have a wide variety of applications, including image classification, object detection, and text classification.

A recurrent neural network (RNN) [96] is a type of neural network that helps in processing

sequential data. RNNs maintain a memory of the previous words in a sentence and use that information to make predictions, thus making use of context. Long-Short Term Memory (LSTM) [53] characterizes a specific type of RNN also capable of handling sequential data. LSTMs specifically address the issue of vanishing gradient by selectively storing or discharging information from previous steps.

An attention mechanism enables a neural network to focus on specific parts of an input sequence when predicting, by determining which words are most relevant in the input. Thus, the output captures better context. Transformers, as introduced by Vaswani et al. in 2017 [112], use an attention mechanism in both the encoder and the decoder.

Transformers have proven to be highly effective in natural language processing tasks such as summarization, question answering, and translation. We can use machine transduction, e.g., transformers, to obtain chapter identifiers in the form of chapter classification labels and summaries from chapter text.

2.4.1 Language Models

Language models help estimate the probability of the next word occurring in a sentence. Thus, they are particularly useful in various natural language processing tasks. There are several types of language models, such as the n-gram models, recurrent neural network (RNN) models, and transformer models. Transformer-based language models have been increasingly used in recent times. Large language models (LLMs – see the next subsection) are being trained on an enormous quantity of data which helps the model better generate contextually meaningful output. Language models can predict correct or relevant terms only when exposed to the domain or language-specific vocabulary. BERT [38], SciBERT [17], RoBERTa [73], and GPT [92] are examples of transformer-based large language models.

BERT is a language model that uses bidirectional training of transformers. It has been trained using the English Wikipedia [113] and an extensive collection of free novels written by unpublished authors known as BookCorpus[2]. Various domain-specific language models have been developed to cater solely to a discipline. BioBERT [63] is a pre-trained model designed to help with biomedical text mining and analysis. SciBERT, on the other hand, was trained on a corpus of 1.14 million computer science and biomedical papers from Semantic Scholar[3]. Unlike BERT-based models that consider the context of the input sequence while predicting the output sequence, GPT is an auto-regressive language model that predicts the next word based on the previous sequence of tokens. In Longformer [18], the authors proposed a 'task motivated global attention' to handle long document tasks. The model was fine-tuned for several downstream tasks like classification, text summarization, and question answering. BigBird Pegasus [117] is another attention-based model that is capable of handling longer contexts. It has also been shown to perform well on several tasks, including summarization.

2.4.2 Large Language Models (LLMs)

Generative AI [48] and LLMs have proven to be effective in many scenarios such as knowledge discovery, answering questions, summarizing research content, and other NLP downstream tasks. Some popular generative LLMs are GPT [92], Llama 2 [110], Llama 3 [7], Gemini [44] and Claude [11]. In [120], the authors surveyed the most popular large language models, covering several characteristics, including performance, training data used, and hardware requirements. Typically, these models are trained on huge real-world text corpora and can be fine-tuned for specific tasks. Although LLM-based systems have been applied to improve the handling of multiple NLP tasks, researchers should be aware of several issues, such as the

[2]https://paperswithcode.com/dataset/bookcorpus
[3]https://www.semanticscholar.org/

fabrication of information associated with using them. Example issues include generating: random text, incorrect statements, misinformation, irrelevant content, texts that promulgate bias (that characterizes the training data), information that violates rules of privacy or security, or stale results (missing fresh information only available after the training date). Finally, not all LLMs are available openly for research. To get the most out of using LLMs, it is imperative to deploy the models locally and pre-train or fine-tune them as per the task and data needs. Figure 2.2 depicts various approaches to training or tuning different language models. Although pre-training can be the most effective way to customize and train any language model, it is rarely feasible to use that approach for LLMs. LLMs have shown to work well with instruction tuning them on specific tasks as discussed in Section 2.4.2.1.

For our work discussed in Chapters 4 and 5, we use Llama 2 and/or Llama 3 as the large language model of choice. Llama 2, released by Meta in July 2023, was trained on trillions of tokens. They have three different parameter models (7B, 13B, and 70B) and two versions for each (chat and non-chat). Llama 2 is openly available for researchers to use through HuggingFace or direct download. Llama 3, on the other hand, was released in April 2024 and contains two models with 8B and 70B parameters. It also has two variations: one is the pre-trained model, and the other is the instruction-tuned model. This 'Instruct' model is specifically trained to be used in a dialog use case and thus is able to learn from instructions provided by the user (few-shot or many-shot learning). This gives us a unique ability to fine-tune the models using instruction tuning on several downstream tasks.

2.4.2.1 Prompt and Instruction Tuning

Prompt tuning, as first discussed in the paper [65], is an effective way to train large language models on downstream tasks. Unlike traditional fine-tuning, with training on specific datasets for the model to work on downstream tasks, prompt engineering involves crafting

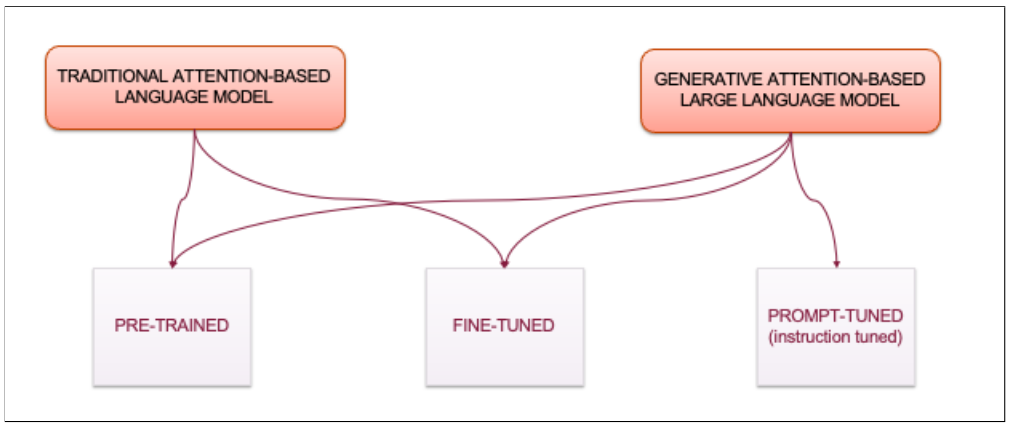


Figure 2.2: Training approaches for language models

user inputs to generate desired results. Prompt tuning refines LLMs by using prompts instead of changing the core parameters of the model. This is effective for LLMs as fine-tuning them is an extremely resource-intensive process. Prompt tuning involves providing prompts to guide the model in generating the desired output for specific tasks. There are multiple approaches for prompt tuning. One can provide prompts and instructions to the model to generate output, similar to zero-shot learning. On the other hand, as in few shot learning, users can provide examples for the model to learn from, alongside the prompts. Although prompt tuning can generate desired outputs for several downstream tasks, it can get challenging when dealing with a specific but large and heterogeneous corpus such as scholarly ETDs.

As discussed above, fine-tuning LLMs is not an option for research settings. Therefore, we resort to instruction tuning. Instruction tuning [115] can be effective when working with specific datasets. It involves changing the model weights as the model is trained on the desired dataset. The resultant model is, therefore, capable of following the instructions provided. Fine-tuning using instructions has been shown to significantly improve the performance of a model when compared to its zero-shot setting [119].

We use LMs and LLMs to generate derived metadata at the chapter level in the form of

chapter classes and summaries. More details are in Chapters 4 and 5.

2.5 Summarization

Summarizing text is a natural language processing task of immense significance for our dataset. There are two classes of techniques for summarization, namely, abstractive and extractive.

Extractive summarization selects one or more portions from the original text for inclusion in the target summary. On the other hand, abstractive methods mimic human techniques, where new words can be used to create a text that covers the semantic content of the original. In [9], the authors give an in-depth description of how extractive summarization works. SummerRuNNer [83] is a recurrent neural network-based tool for extractive summarization. BERTSumm [72] utilizes the BERT language model to perform extractive summarization. The authors used BERT and fine-tuned it by adding summarization layers on top of it. Some state-of-the-art methods for abstractive summarization include the Pointer Generator [100] and Fast Abstractive Summarization with Reinforce-Selected Sentence Rewriting [29]. The Pointer Generator is a sequence-to-sequence model capable of generating new words related to the original text. In BERT-AL [118], the authors propose a model to handle long sequences by introducing a multi-layer LSTM into each layer of BERT. The authors evaluated BERT-AL using the CNN/Dailymail summarization dataset [52], which is a dataset with an average token size of 781. In [54], the authors propose a new attention mechanism to gather salient content from input tokens and extend the context length to up to 10K tokens. In [24], the authors use the document's structure to aid in the summarization process by creating a document structure tree. The authors annotate a dataset that depicts the question and summary hierarchy from long documents. Therefore, given a question (context), their model

is trained to generate a summary. The Longformer encoder-decoder (LED) architecture is capable of handling up to 16K tokens at a time. The authors also evaluated various benchmark datasets for each of the downstream tasks. Research proposed in [62] introduced a hybrid approach that includes extractive and abstractive methods. Most of these works use either a named entity recognition tool or some form of information extraction tool [50, 68].

2.6 Systems and Services

As a subject matter expert in CS5604's Fall 2022 and 2023 offerings, I mentored a group of 4-5 students in each semester. Our first iteration of the prototype system developed in the Fall 2022 offering of CS5604 [37, 43, 58, 87, 103] was comprised of services supporting the needs of both readers and developers (experimenters). The services developed displayed classification results of already segmented documents on the document view page. It also enabled developers to experiment with classification models by uploading a document and selecting the model they wanted to use on the experimenter page. This work was published and co-presented at ACM/IEEE JCDL [27] in 2023. The team working on classification and summarization in Fall 2023 [82] engaged in enhancing and improving the system further. This work involved using LLMs in the workflow, generating classification labels and summaries for more documents, and improving collaboration with other teams in the course (with better APIs and organization of system components).

2.7 Evaluation

We employ some commonly used metrics – such as accuracy, precision, recall, and F1 measure – to report the performance of our machine learning models. We discuss BLEU [88] and

ROUGE [67] metrics to evaluate automatically generated summaries.

First, we define some of the terms that help us calculate the aforementioned metrics.

True Positive (TP) reflects the positive classes that have been classified as such.

True Negative (TN) reflects the negative classes that have been classified as such.

False Positive (FP) reflects the classes erroneously classified as positive.

False Negative (FN) reflects the classes erroneously classified as negative.

2.7.1 Accuracy

Accuracy, as shown in Equation 2.1, is the measure of correctly classified data instances among the total number of predictions. Accuracy tends not to be the best measure when data is imbalanced.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

2.7.1.1 Precision

Precision, as shown in Equation 2.2, is a measure of true positives amongst all the predicted positive instances. A high precision (close to 1) value indicates that the classifier is doing a good job, wherein when a class is assigned, it tends to be correct.

$$Precision = \frac{(TP)}{(TP + FP)} \tag{2.2}$$

2.7.1.2 Recall

Recall, as shown in Equation 2.3, is a measure of sensitivity or true positive rate. It records the true positives among all actual positive instances. Thus it indicates how many of the members of a class are identified.

$$Recall = \frac{(TP)}{(TP + FN)} \tag{2.3}$$

2.7.1.3 F1 Measure

F1 measure, as denoted in Equation 2.4, is a balanced metric. It is calculated by finding the harmonic mean of Precision and Recall. F1 score is a better metric than accuracy when dealing with an imbalanced dataset.

$$F1 = \frac{(2 \times Precision \times Recall)}{(Precision + Recall)} \tag{2.4}$$

2.7.1.4 Micro, Macro, and Weighted Averaging

Precision, Recall, and F1-scores have micro, macro, and weighted averaging of the metrics.

- Micro-averaging is used to provide the overall performance by giving equal weightage to each instance. This is well suited for a balanced dataset.
- Macro-averaging is used to understand the model performance for each class, especially when the data is imbalanced. This metric can help understand the performance of minority classes better.
- Weighted-averaging adjusts for class imbalance issues by adjusting the weights.

2.7.1.5 ROC Curve

The Receiver Operative Characteristic (ROC) curve is a graphical plot describing classifier performance at all thresholds. It was first developed for use in signal detection theory in the 1940s. However, it is now used in a variety of fields such as medicine, epidemiology, statistics, and machine learning [20, 34, 51]. The ROC curve shows the true positive rate (sensitivity) on the y-axis, and the false positive rate (1 - specificity) on the x-axis. The true positive rate is the proportion of actual positive cases that are correctly identified as such, while the false positive rate is the proportion of actual negative cases that are incorrectly identified as positive. A diagonal line from the bottom-left corner to the top-right corner represents the ROC curve of a random classifier, with an area under the curve (AUC) of 0.5. The closer the ROC curve is to the top-left corner, the better is the classifier's performance. ROC curves are useful for comparing the performance of different classifiers and for selecting the best threshold for a given classifier, depending on the relative importance of false positives and false negatives in a particular application.

2.7.2 Summarization Evaluation

2.7.2.1 BLEU Score

Bilingual Evaluation Understudy [88] or BLEU scores were developed in 2002 to evaluate machine translation tasks. However, they are also used in evaluating text summarization. BLEU scores are calculated by using a modified n-gram precision, for various values of n. They measure how many n-grams in the candidate sentences match the reference sentences. While BLEU is very effective in detecting extracted phrases and matches, it falls short when paraphrasing or synonyms are involved. For machine translation tasks, precision or exact matches between the candidate and the reference statement are important. However, this

is less relevant when considering summarization. The score favors summaries when exact sentences are picked from the input text in the generated summary. This is often not the case, especially for abstractive summarization techniques. Therefore, BLEU scores are not the appropriate metric to evaluate automatically generated abstractive summaries.

2.7.2.2 ROUGE Metric

The 'Recall-Oriented Understudy for Gisting Evaluation' (ROUGE) set of scores [67] is commonly used to evaluate machine translation and summarization tasks. ROUGE scores help us evaluate an automatically generated summary against a reference summary, often called the gold (standard) summary. The most common ROUGE metrics are as follows.

- ROUGE-N measures the overlap of n-grams between the automatically generated and gold summaries.
- ROUGE-L measures the longest common subsequence between the automatically generated and gold summaries.
- ROUGE-S measures the skip-bigram co-occurrence. This allows us to search for words occurring consecutively in the gold summary but that might be separated by word(s) in the generated summary.

All ROUGE metrics report the F1-Score, Precision, and Recall.

2.7.2.3 G-Eval

Evaluating summaries poses several problems. In order to evaluate summaries, we need ground truth reference summaries to compare with automatically generated summaries.

In [71], the authors suggest evaluation using LLMs to evaluate natural language generation tasks such as summarization. The authors proposed a framework for evaluating natural language generation (NLG) outputs using an LLM and the chain-of-thought (CoT) approach. The research uses prompting on GPT-4 to generate scores for coherence, consistency, fluency, and relevance. For summarization evaluation, the authors used the SummEval dataset to evaluate this approach. It was shown that the proposed evaluation method, named G-Eval [71], surpassed all other state-of-the-art evaluators on the SummEval dataset.

2.7.3 Human Evaluation

Apart from obtaining automatic evaluation scores for text, it is imperative that we conduct a human evaluation to evaluate the quality of the generated summaries. Prior research has shown that leveraging human capabilities helps improve AI models. In [25], the authors use human annotators for training data. They reported from their study that annotators did not need to be experts. In [64], the authors conducted a study to evaluate the human and AI model interaction, where interactivity leads to a richer evaluation. For example, a user is asked to edit an automatically generated summary or find answers to questions by querying the model. In [116], the authors proposed a method to leverage human feedback to learn from and thus create better summaries. They used the task decomposition method to train the model on smaller parts. They used human feedback to improve on bigger tasks.

Chapter 3

Datasets: Curation and Preparation

3.1 Chapter Overview

In this chapter, we start by discussing our ETD repository of over half a million documents. As a part of the IMLS-funded research collaboration with ODU [55], we have amassed a collection of ETDs from across the United States, covering many disciplines. ETDs are typically hosted by university repositories and stored and shared as PDF files. We harvested the ETD full text and other metadata information in these repositories. Long ETDs need to be segmented into chapters to perform chapter-level analysis. We discuss some of the automatic segmentation approaches and analyze the results from those methods. We introduce all of our data and subsets used in various experiments discussed in the research work in the dissertation. Finally, we discuss a text extraction and data pre-processing pipeline that is useful to obtain 'clean text' from an ETD and its chapters.

3.2 Dataset Collection and Analysis

We have amassed a collection of over 500,000 ETDs across academic institutions in the United States. This effort has been reported and published in [111]. Our ETD dataset comprises born-digital and scanned PDF files. Born-digital documents have been converted to PDF digitally with the help of software such as Word or LaTeX. Most PDF files since

Table 3.1: Metadata field distribution

Metadata Field	Count	Description
id	533047	Unique ID attached to each document in our collection
title	533042	Document title
author	532957	Author name
advisor	416513	Name of committee advisor
year	467088	Year the document was made available
abstract	433688	Author provided abstract
university	522976	University name
degree	417683	Indicating masters' or doctoral document
URI	532875	Identifier for the document
department	288502	Department information [see Section 3.2.2]
discipline	365519	Discipline information [see Section 3.2.2]
language	489372	Language of the document
schooltype	458033	Identifies if a school is an HBCU, HSI, or regular

the year 2000 have been born-digital. We had 339,485 born-digital and 170,217 scanned documents in our collection as of February 2022.

Table 3.1 depicts the available metadata fields in our repository. We see that not all metadata fields in the repository have information in them for all ETDs.

The number of ETDs in a university repository is determined by the number of theses and dissertations generated at that university and the percentage of those works included in that repository. Figure 3.1 depicts the distribution of ETDs from 15 US universities with the largest ETD repositories that we have crawled. These repositories encompass over 53% of the total number of documents in our collection.

3.2.1 HBCU and HSI

In our collection of ETDs, we strive to make the dataset representative. Apart from diversity in disciplines, we harvested documents from several HBCU and HSI colleges or universities [31]. Tables 3.2 and 3.3 represent documents from HBCU and HSI sites that exist in

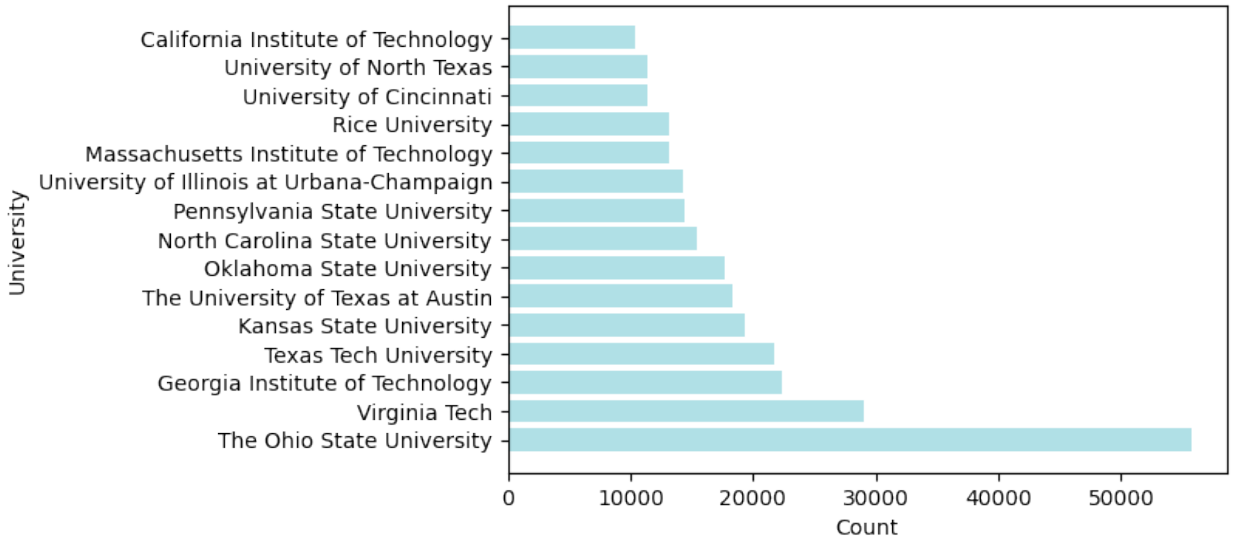


Figure 3.1: Distribution of ETDs by University

our repository. We have a total of 85828 (15%) HSI and 1964 (0.3%) HBCU documents. We recognize that the total number of documents from these sites compared to the collection size is small. We have plans for increasing this number in the future.

3.2.2 Discussion

In this section, we go over some of the unique features of our dataset.

1. A good percentage of the metadata fields are empty, as shown in Table 3.1. Most of the metadata information is optional for the author to supply, thus resulting in empty fields.

2. A discipline typically refers to an area of research or study, whereas a department refers to the academic branch in the institute that offers courses and research in a discipline. Many universities don't follow the traditional definition of department and discipline and use the terms interchangeably.

Table 3.2: Documents from HSI Sites

University	Count
The University of Texas at Austin	23184
Texas Tech University	21781
Texas A&M University	12817
University of North Texas	11411
Florida International University	3672
University of California, Riverside	3458
University of California, Irvine	3083
University of California, Santa Cruz	1792
University of California, Santa Barbara	1731
Texas State University	1678
Texas State University, San Marcos	642
University of California, Merced	579
Total	85828

Table 3.3: Documents from HBCU Sites

University	Count
Morgan State University	596
Howard University	394
South Carolina State University	302
Tennessee State University	219
Delaware State University	151
Bowie State University	91
Grambling State University	87
Virginia Union University	45
Jackson State University	43
University of Arkansas at Pine Buff	28
Norfolk State University	7
Kentucky State University	1
Total	1964

3. Metadata information is typically filled into a text box by the author and not selected from a drop-down menu. This results in inconsistencies, spelling mistakes, and typographical errors.

Missing metadata can lead to data imbalance problems, especially for classification tasks. Most of the inconsistencies require additional data cleaning and pre-processing steps. These are discussed in detail with regard to each of the downstream tasks discussed henceforth.

3.3 Segmentation

In this section, we go over the approaches for chapter segmentation. ETDs are long documents, often 70-100 pages long; some are much longer. Table 3.4 shows the count of pages in our ETD repository. We see that over 300,000 (over 50%) documents are 100 to 500 pages in length. The average page count of ETDs in our current repository is 144.8 pages, with a median value of 127. We need to segment such long documents into chapters before we can proceed with classifying and summarizing at that finer level of granularity. Identifying chapter boundaries correctly from the long documents is hard to do automatically. In the following subsections, we discuss approaches to automatically segment chapters of ETDs.

Table 3.4: Page Count of ETDs

Number of Pages	Count of Documents
5-9	637
10-99	178,986
100-499	324,746
500-999	2908
Above 1000	149

3.3.1 Segmentation using LaTeX

This approach involves using LaTeX source text files to identify chapter boundaries of ETDs. arXiv [46] is an open-access scholarly preprint server mainly catering to STEM disciplines – specifically engineering and natural sciences. arXiv requires authors to submit the document source text files alongside the PDFs. Therefore, we have the LaTeX sources available for use. arXiv has a small subset of ETDs, and we use the sources from those ETDs to create a segmentation model. The segmentation model uses page text and images to predict chapter boundaries. This work and the dataset have been reported in [78] and [57].

We looked at a subset of segmented results to determine the performance. A significant number of chapters were missed. Many chapter boundaries were not detected correctly. A possible reason for such problems is that the model was trained on LaTeX documents from arXiv. The arXiv service caters to a very specific set of STEM domains, ignoring many of the domains found in our corpus. Further, only a very limited number of the works in arXiv are ETDs. Hence, while training, the model was exposed to a very small amount of data, with few document variations, taken only from a small number of STEM-based disciplines. Therefore, it fails to perform well on a corpus very different from arXiv. See Tables A.1 and A.2 for the performance of the 'Chapter Start' label from non-arXiv documents using two approach variants. Though the F1 score for 'Chapter Start' shown in Table A.1 was higher than that shown in Table A.2, it still was only 46%, making clear there were many errors.

3.3.2 Segmentation using Object Detection

We modified the object detection-based work by Ahuja et al. [5] to segment chapters. The model takes an ETD PDF and outputs an XML file with elements marked up. The XML tags are depicted in Table 3.5. This Yolo-based [114] object detection model was used to

generate XML files for a randomly sampled set of 3000 documents taken from our ETD repository. These XMLs were generated in the Fall of 2022 by Team 3 from CS 5604 [37]. The average time per document was 4 minutes, so it took over a week to process these 3000 documents.

Table 3.5: Tags in the generated XML

etd	front	title	author	university	degree	committee
date	abs_heading	abs_text	tocs	toc	toc_text	body
chapter	title	sections	section	name	paragraphs	para
figures	figure	tables	table	equations	equation	algorithms
caption	footnote	footnotes	back	ref_heading	ref_text	path

We use the Python ElementTree [76] parser to extract the chapters. The 'chapter' tag indicates the beginning of each chapter. The 'chapter' tag has nested tags as depicted in Figure 3.2. We extract the text from all of the sections and their paragraphs to create separate chapter-specific text files. This XML method of segmenting chapters enables us to separately extract text, as well as elements like figures, captions, and equations.

As we analyzed the XMLs, we came across some interesting anomalies. From Figure 3.3, we see that over 1200 documents had 6-10 chapters. Also, over 1000 documents had more than 10 chapters. It is unlikely that a single ETD will have greater than ten chapters. We studied a few PDFs and their XMLs to identify the cause of such errors. For a particular document, we discovered there were 17 chapters generated, whereas the document, in reality, had 4 chapters. This error was due to the model falsely recognizing sections within a chapter as a chapter boundary. We found that the most common error was to consider the abbreviations, definitions, abstracts, and acknowledgments as chapters. Some subsections and page headers are also marked as a chapter beginning, further leading to the incorrect prediction of a large number of chapters.

```
        <chapter> ⋯ </chapter>
        <chapter> ⋯ </chapter>
        <chapter>
            <title>Chapter One
Introduction: Writing Postdramatic Theater</title>
            <sections>
                <section>
                    <name>Defining the Postdramatic</name>
                    <paragraphs>
                        <para>ESTRAGON: [gesture towards the universe] This one is enough for you?
(Samuel Beckett, Waiting for Godot)
In his 1999 book Certain Fragments, director Tim Etchells describes “the spectacle of
‘new playwrights’ at a 1997 conference in London’s Royal Court Theatre”: the writers’ “biggest
(almost only) topic of conversation seemed to be long pontifications on the understanding of a
comma” (104). “Hard for me to understand,” he continues drily, “having never much cared for
punctuation…. Never cared much for playwrights” (104–5).
Etchells is the founder and artistic director of Forced Entertainment, the UK’s highest-
profile experimental theater company. He is also a writer, and the essay that contains these
remarks, “On Performance Writing,” expounds on the kind of text that might ultimately make the
pedantic “playwrights” obsolete: this is theater’s newfound “gabbling voice composed of scraps
and layers, fragments, quotations” (99). In this vein, the essay makes a case for the theatrical
power of writing. And yet it dismisses the playwrights for taking the writing of writing, its
literary mechanics, too much to heart: “How directors and actors can’t understand a comma these
days. The terrible shame of it” (104). The implication, of course, is that directors and actors have
more important, more exciting things to worry about. If a writer wants to hang with theater’s
advanced guard, she’d better shake this stodgy graphophilia, put down the MLA guide and get
into the game.
Etchells’s work with Forced Entertainment exemplifies the formally and conceptually
innovative theater that has increasingly become known as “postdramatic.” His rejection of the
playwright would seem to fit neatly with this label; after all, playwrights are also called
“dramatists.” And in fact, the critical discourse on postdramatic theater often assumes that a
dethronement of the once-dominant playtext marks the crucial divide between the old theater and
the new. The present study attempts to refute this theatrical common sense. I will argue for a
concept of the postdramatic that not only includes such punctuation-obsessed playwrights as
Gertrude Stein and Suzan-Lori Parks, but needs them; a postdramatic theater for which literary
work is not one theatrical “material” among others, but a privileged mode of enacting theater’s
most urgent project. Briefly, this theater is one of heightened negativity: a specifically utopian
response to the heightened actuality that has often seemed to distinguish performance from other
kinds of art. I will explain this formulation in what follows, grounding it in Theodor W.
Adorno’s critical theory, and suggesting that it structures a series of modernist and contemporary
texts. But I want to begin by reviewing the scholarly discourse that has arisen around the idea of
the postdramatic, and clarifying my own relationship to this discourse.</para>
```

Figure 3.2: XML structure. Adapted from [5].

3.4 Segmented ETD Dataset (ETD-SGT)

We discuss segmentation approaches and problems associated with each of the approaches (that are explained in Section 3.3). Models can help to automatically segment ETDs, but the performance varies greatly depending on the style and discipline of the ETD. Nevertheless, we need an accurately segmented ETD dataset that will be used in subsequent work to create chapter identifiers, classification labels, and summaries. Therefore, in this section,

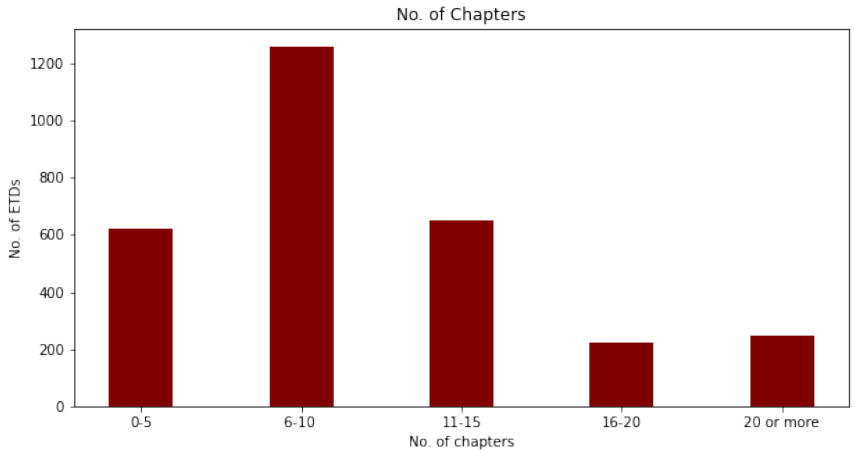


Figure 3.3: Distribution of chapter numbers in ETDs

we introduce the ETD-SGT (ETD Segmented) dataset. The ETD-SGT dataset is a ground truth dataset of chapters manually segmented from full-text ETDs. The documents were selected from our ETD repository of over 500,000 documents. We applied the following criteria for selecting the documents:

- We start by looking at the count of documents per university. We select the top 10 universities by count.
- We then select the top disciplines/departments by counting the number of documents from the top 10 universities.
- From the disciplines identified, we identify a list of STEM and non-STEM disciplines.
- We select disciplines from this STEM and non-STEM list and add documents for manual segmentation for each iteration of ETD-SGT-*.

We segment each of the ETDs according to the following scheme.

- Pages before the first chapter are named as in the **front**.

- Each chapter is named as **chapter<i>**.
- Bibliographies are extracted and named as **references**.
- Any appendix available is also extracted as a separate PDF file named **appendix**.

Some of the uses of the above-mentioned types of segments are as follows.

- The front matter, including the abstract, can be used as an additional source of features for any downstream task. An example is noted in Chapter 5.
- Each of the chapters can be used for downstream tasks like chapter-level classification and summarization, as described in Chapters 4 and 5, respectively.
- References can help create citation graphs. We can use reference information such as the titles of references to help gain context while generating chapter summaries as discussed in Chapter 5.

3.4.1 ETD-SGT-1

The first version of the ETD-SGT dataset comprises documents from both STEM and non-STEM fields, coming from departments with a significant count in our ETD repository of over half a million documents. We chose the following disciplines based on the count of documents, considering two criteria:

1. Universities with the largest number of document counts (as per Figure 3.1)
2. A mix of STEM/Non-STEM disciplines was selected on the basis of the largest number of documents.

The 11 departments in this dataset are Biology, Computer Science, Mechanical Engineering, Psychology, History, Business Administration, Architecture, Electrical Engineering, Public Policy, English, and Education. Table 3.6 shows some statistics of the segmented dataset. We started with a subset of 200 documents, but we found 8 ETDs unusable as they were missing the original ETD PDF.

Table 3.6: ETD-SGT-1

Feature	Count
Number of Documents	192
Number of Chapters	1074
Number of Departments	11

3.4.2 ETD-SGT-2

Employing time-intensive manual segmentation methods, we gradually and continually increase the number of documents in the ETD-SGT dataset. Thus we started at 10 and went up to 50. The selection policy of the documents remains the same as ETD-SGT-1 making the current document number 242. The approach for segmenting is the same as the one used in ETD-SGT-1, as shown in Section 3.4.1.

3.4.3 ETD-SGT-3

Our VT-wide user study is discussed in Section 6.3.3.3. We received interest from participants from a diverse set of disciplines. Some of the disciplines already existed in our ETD-SGT-* corpus. However, we were fortunate to attract interest from domains such as Neuroscience and Ecology – domains that did not exist in our ETD-SGT-* corpus. Therefore, we added

ETDs from these domains to the ETD-SGT set . The final ETD-SGT dataset during the time of this ETD submission has 244 documents. The approach for segmenting is the same as the one used in ETD-SGT-1, as shown in Section 3.4.1.

3.5 ETD Data Subsets

3.5.1 PQDT

A dataset of 9298 documents across 28 different subject categories mapping to the ProQuest subject category system was used and reported in [59]. The number of categories and documents was limited to 28 and 9298, respectively, to balance the dataset yet have enough data to train machine learning and deep learning models. At most, each subject category has 500 documents. Table 3.7 shows the distribution for that dataset of documents across various categories.

We notice that minority classes, like Forestry and Industrial Engineering, comprising 121 and 196 documents, respectively, indicate that the dataset is not balanced. We also note that there are 17 STEM disciplines with a total of 6734 documents and 11 Non-STEM disciplines with 5198 documents. Therefore, the dataset is not distributed evenly among STEM and non-STEM fields. The subtotals above and the entries in the table include documents that might belong to multiple subject categories. However, we have 9298 unique ETDs that were used in the final dataset.

Table 3.7: Number of ETDs in each subject category in the PQDT ETD collection (alphabetic order). Adapted from [59].

Subject Category	**Number of ETDs**
Adult education	500
Aerospace engineering	386
Biomedical engineering	485
Chemical engineering	325
Civil engineering	496
Computer Engineering	380
Computer science	500
Ecology	500
Educational leadership	500
Educational psychology	500
Electrical engineering	500
Elementary education	499
Environmental science	494
Forestry	121
Higher education	500
Industrial engineering	196
Marketing	271
Materials science	482
Mathematics	222
Mechanical engineering	500
Molecular biology	500
Occupational psychology	500
Organic chemistry	255
Public administration	428
Secondary education	500
Special education	500
Statistics	392
Teacher education	500
Total	11932

3.5.2 ETD-CL

The ETD-CL subset has been curated specifically for the classification task from our ETD repository. It covers a set of 47 disciplines balanced across STEM and non-STEM fields.

To verify that a department belongs to STEM, we use the U.S. Department of Homeland Security (DHS) list[1]. This dataset has been manually curated to have a consistent identification of discipline that is guaranteed to eliminate the inaccuracy of metadata, as reported in Section 3.2.2. We have applied the following steps to curate this dataset:

- We start by looking at the disciplines and department information in our metadata. Universities tend to use these terms interchangeably. Thus, we need to look at both fields to gather as much data as possible.
- We use lowercase, strip any spaces, and use a spell checker to correct the metadata.
- We omit superfluous words. Thus, the "department of physics" and "physics department" are represented by "physics".
- After cleaning the metadata fields, we sort the discipline information by count of documents. Our goal is to obtain a balanced dataset evenly spread across STEM and non-STEM disciplines.
- We select the top 25 STEM and non-STEM disciplines (each) to have enough diversity in the document set and be representative of our ETD repository.

Tables 3.8 and 3.9 list the 25 STEM and 22 non-STEM disciplines. The departments are listed in descending order of the count of the total number of ETDs available. We had to drop 3 non-STEM disciplines as these three disciplines had fewer documents than 200. To have a balanced set with a considerable number of documents for training, we make sure that each discipline has exactly 200 documents.

Department names are left in the form provided in the metadata. Though it might be helpful to normalize or group them, there are dangers in doing so, and such work is beyond the scope

[1]https://www.ice.gov/sites/default/files/documents/stem-list.pdf

of this study.

Table 3.8: STEM ETDs

Department
Aerospace engineering and environmental engineering
Animal science
Architecture
Biology
Biomedical engineering
Chemical engineering
Chemistry
Civil, architectural,
Communication sciences and disorders
Computer science
Crop science
Earth and atmosphere sciences
Economics
Geological sciences
Home economics education
Kinesiology and health education
Linguistics
Materials science and engineering
Mathematics
Mechanical engineering
Petroleum and geosystems engineering
Physics
Political science
Psychology
Electrical and computer engineering

Table 3.9: Non-STEM ETDs

Department
Advertising
Anthropology
Art design and history
Business administration
Communication studies
Community and regional planning
Counselling leadership, adult education and school psychology
Curriculum and instruction
Education
English
Geography
Government
History
Journalism
Music
Philosophy
Public affairs and policy
Radio-television-film
Social work
Sociology
Spanish, Italian and Portuguese
Theatre and dance

3.5.3 Dataset for Fine-tuning Language Models (FTD)

In this section, we describe the data subset that was used to fine-tune language models in Section 4.3.2.2. We use ETDs from our ETD repository as discussed in Section 3.2. The documents comprise two University of California schools: Berkeley and Irvine. We have

taken all the documents from these two universities. This set is thus diverse across various disciplines as both California state schools cater to a wide variety of subjects. We only use born-digital documents to fine-tune language models, as text extracted from scanned documents often results in noisy data. Statistics about the fine-tuned language models with and without front matter is discussed in detail in Table 3.10. We use the full text (with and without front matter) from these ETDs to fine-tune language models as discussed in detail in Section 4.4.2.2.

Table 3.10: Statistics of ETDs used for fine-tuning language model (for ETD-FTD set)

ETD feature	**Mean Word Count**
Full-text	51154
Without front matter	45792

3.5.4 ETD-SummEval

To evaluate automatic summaries, we need to have gold standard summaries to compare against. While a document abstract is commonly available, chapter abstracts are rarely found. Thus, evaluating such summaries becomes a significant challenge. We have manually selected ETDs that have a summary or an abstract associated with each chapter. While this rarely occurs, we still have found a couple of universities where a summary or a discussion section is associated with each chapter of an ETD. An alternative to this is to find a scholarly article that corresponds to an ETD chapter, where the article has an abstract. This situation arises for some of the chapters of ETDs that are directly related to a conference proceeding paper or a journal article. This gives us the abstract of the peer-reviewed article as the gold standard summary. However, without the paper attached to a chapter, we need to

search the author's Google Scholar listing to find a corresponding paper, e.g., with a title similar to a chapter name. In yet another alternative approach, we need to apply an object detection model that will identify an "abstract" or "summary" section at the start of a chapter. The three aforementioned techniques are extremely time-intensive, requiring careful manual intervention. Thus, we have only been able to curate a small collection of chapters and their abstracts. We keep incrementally adding more chapters to this list as we come across more documents with abstracts that are associated with chapters. We have only 50 such ETD chapters with summaries. They belong to the following disciplines: English, biology, mechanical engineering, education, and computer science.

3.5.5 ETD Dataset Description and Uses

Table 3.11 provides a summary and some statistics regarding the ETD data subsets discussed in the previous subsections.

3.6 Text Extraction and Pre-processing

Each ETD comprises several chapters. We perform text extraction on each of the documents and chapters (where segmented, such as from the ETD-SGT* datasets). We compare various Python-based tools (as discussed in Chapter 2, Section 2.1) for this task. We discuss the text extraction approaches and our observation in Table 3.12.

PdfPlumber [104] performed the best; thus, our first approach was to use it to extract text from the PDFs. Figure 3.4 shows how the text from equations is extracted. We see that most of the mathematical relationships are not extracted correctly. Such input data is not very helpful for tasks such as fine-tuning language models, classifying, and summarizing content.

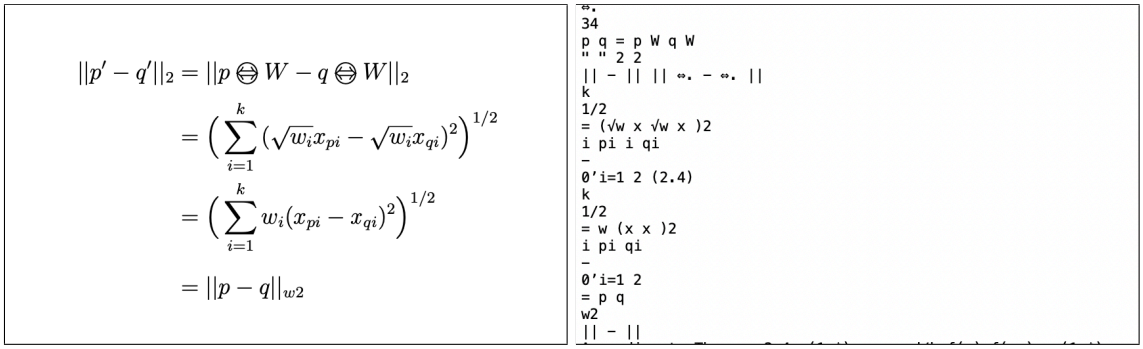
Table 3.11: ETD data subsets

Dataset	Description	No. of Documents	Task (Section)
ETD-SGT-1 ETD-SGT-2 ETD-SGT-3	Manually segmented documents	192 50 2	Classification, Summarization (3.4)
PQDT	Curated from ProQuest ETDs [59]	9298	Classification (3.5.1)
ETD-CL	Manually curated	9400	Classification (3.5.2)
FTD	Born-digital ETDs	8200	Data used in fine-tuning language models (3.5.3)
ETD-SummEval	Born-digital ETDs	45 chapters with ground truth summaries	Data used to set up summarization baseliness (5.3.1)

Therefore, we remove such elements as an extra step to help improve the utility of our output text. We leverage the work done in [5] to remove the unwanted elements in the generated text. Figure 3.5 is a list of elements we deem not helpful. Accordingly, we decided to exclude them from the extracted text of the ETDs.

Table 3.12: Comparison of Text Extraction Tools

Name	Overall Quality	Comments
Abbyy Cloud OCR SDK	Very Good	Paid
PDFMiner	Moderate	Many blank extractions observed
PDFPlumber	Good	Easily scalable
PyMuPDF	Good	Not easily scalable
AWS Textract	Excellent	Paid



$$
\begin{aligned}
||p' - q'||_2 &= ||p \ominus W - q \ominus W||_2 \\
&= \Big(\sum_{i=1}^{k} (\sqrt{w_i}x_{pi} - \sqrt{w_i}x_{qi})^2\Big)^{1/2} \\
&= \Big(\sum_{i=1}^{k} w_i(x_{pi} - x_{qi})^2\Big)^{1/2} \\
&= ||p - q||_{w2}
\end{aligned}
$$

```
34
p q = p W q W
" " 2 2
|| - || || ∞. - ∞. ||
k
1/2
= (√w x √w x )2
i pi i qi
-
0'i=1 2 (2.4)
k
1/2
= w (x x )2
i pi qi
-
0'i=1 2
= p q
w2
|| - ||
```

Figure 3.4: Equation and its extracted-text version

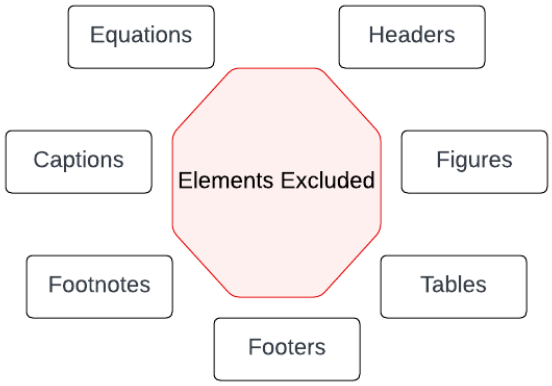


Figure 3.5: Elements excluded from extracted text

3.6.1 Ensemble Text Extraction Pipeline

Earlier in this section, we described text extraction using PdfPlumber. We also discuss approaches to remove unwanted elements such as figure captions, etc. However, manual evaluation proved that Pdfplumber, alongside object detection approaches, was not performing the best. The extracted text was often noisy – thus warranting a different approach. This section discusses our second approach, i.e., text extraction using a hybrid method of AWS Textract [101] and the object detection methods described in [5]. AWS Textract has been shown to perform well in text extraction, especially from born-digital documents. Figure 3.6 depicts the ensemble pipeline that uses bounding box information from each of the AWS

Textract and object detection methods to extract clean text. This text extraction approach incurs cost. Therefore, we have only extracted text from documents in the ETD-SGT* datasets, which we later use for our downstream tasks.

In Step 1, we perform the following to extract text from ETD pages using AWS Textract.

1. We first create page images in JPEG format from each page of an ETD. This step is required if Textract is used through AWS access keys and APIs.

2. We pass the page images sequentially to Textract by using the "detect_document_text" method.

3. We save the responses in JSON format for parsing.

Figure 3.7 shows the elements extracted from a page using Textract. Three different 'Block-Types' are detected by Textract: PAGE, LINE, and WORD. Each 'BlockType' that is a LINE or a WORD has a confidence score, text, and several other extracted attributes, as shown in Figure 3.7. We specifically look at the extracted text and the bounding box information. We create a script that parses JSON files and extracts the relevant information.

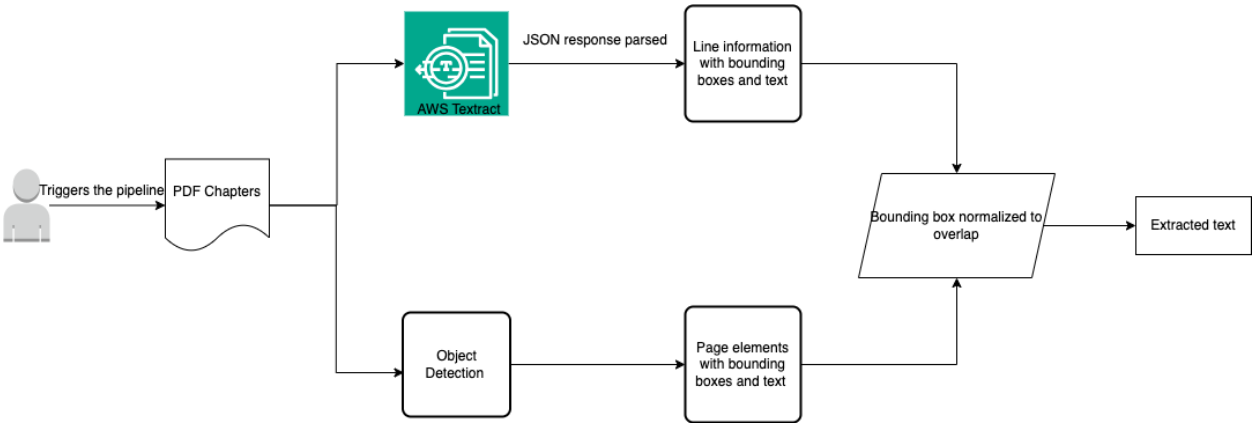


Figure 3.6: Ensemble text extraction pipeline

```
{
    "DocumentMetadata": {
        "Pages": 1
    },
    "Blocks": [
        {
            "BlockType": "PAGE",
            "Geometry": {…
            },
            "Id": "bfc805b5-848f-4b41-87a8-0d6d6788234a",
            "Relationships": […
            ]
        },
        {
            "BlockType": "LINE",
            "Confidence": 99.85183715820312,
            "Text": "Chapter 1",
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.19547615945339203,
                    "Height": 0.02908618189394474,
                    "Left": 0.11925502121448517,
                    "Top": 0.23922903835773468
                },
                "Polygon": [
                    {
                        "X": 0.11925502121448517,
                        "Y": 0.2395031601190567
                    },
                    {
                        "X": 0.3146960437297821,
                        "Y": 0.23922903835773468
                    },
                    {
                        "X": 0.3147311806678772,
                        "Y": 0.2680412530899048
                    },
                    {
                        "X": 0.11928840726613998,
                        "Y": 0.26831522583961487
                    }
                ]
            },
            "Id": "128416f5-8b0a-4a38-a58b-2025b867b806",
            "Relationships": […
            ]
```


Figure 3.7: AWS Textract JSON response

In Step 2, we generate bounding boxes using the object detection method for each page of the ETD. Figure 3.8 shows the bounding box information extracted. The model generated the four coordinates along with the label and page numbers.

In Step 3, we use the bounding box information from both AWS Textract and object de-

```
xmin,ymin,xmax,ymax,label,page_no
182.78280639648438,1503.400634765625,1517.01806640625,1959.748046875,paragraph,0
186.85360717773438,1062.2269287109375,1515.8394775390625,1379.8260498046875,paragraph,0
188.31861877441406,978.429931640625,868.8055419921875,1051.060302734375,Chapter subheading,0
179.83140563964844,869.7340698242188,767.62158203125,951.9418334960938,Chapter subheading,0
170.9706268310547,499.5916442871094,654.5276489257812,769.8641357421875,Chapter Title,0
193.04090881347656,1419.9683837890625,1343.709716796875,1491.7821044921875,Chapter subheading,0
827.642578125,2056.038330078125,866.10400390625,2093.888916015625,page number,0
184.24212646484375,1022.6649780273438,1518.6177978515625,1558.3499755859375,paragraph,1
181.23777770996094,378.9232482910156,1522.6689453125,897.3026733398438,paragraph,1
190.70375061035156,1681.8912353515625,1511.8311767578125,1825.3939208984375,paragraph,1
188.58106994628906,1595.1607666015625,1176.2685546875,1667.382080078125,Chapter subheading,1
184.3738555908203,938.1892700195312,912.405029296875,1008.13916015625,Chapter subheading,1
178.59266662597656,281.5631103515625,665.0501708984375,360.1454162597656,Chapter subheading,1
197.83424377441406,1831.641845703125,1509.7933349609375,2025.2080078125,foot note,1
732.7305297851562,185.3880615234375,1146.0574951171875,256.22357177734375,Chapter Title,1
182.10418701171875,182.05650329589844,1152.9720458984375,254.99490356445312,Chapter Title,1
1470.7032470703125,203.4036102294922,1507.5743408203125,241.36019897460938,page number,1
183.6009521484375,603.9297485351562,1522.2852783203125,1122.502685546875,paragraph,2
183.0892791748047,1250.383544921875,1520.3681640625,1987.5533447265625,paragraph,2
180.37808227539062,281.57147216796875,1521.8968505859375,487.6209411621094,paragraph,2
189.47857666015625,520.8218383789062,1321.8470458984375,595.99365234375,Chapter subheading,2
195.46356201171875,1159.3009033203125,1501.386962890625,1236.21826171875,Chapter subheading,2
```

Figure 3.8: Bounding box information from object detection

tection to weed out the unwanted elements as listed in Figure 3.5. It is crucial to note that the bounding box in both the approaches does not match regarding having the same scale. Therefore, to make sure we can compare bounding box information, we need to scale up/down the information for one of the coordinates. Accordingly, we scale up the bounding box information from Textract by multiplying X and Y coordinates by 1700 and 2200, respectively. We chose these numbers as the YOLO-V8-based object detection method works on page images of size 1700*2200. Therefore, we are left with clean text from each of the chapters of the ETD-SGT* datasets.

3.7 Conclusion

In this chapter, we review our primary ETD repository and other ETD subsets used throughout the research work. We discuss several automatic segmentation approaches and arrive at the following conclusions for the hypothesis:

"Our approach to automatic chapter boundary detection using a neural network model trained on long scholarly documents yields better results than available SOTA neural network models trained on collections of shorter documents."

1. We find that the neural model trained on scholarly documents yields better results than SOTA models trained on shorter documents, as shown in Section 3.3.2.

2. The neural model generates false positives for chapter boundaries. Therefore, the quality is unacceptable. Hence, we rely on manual segmentation to create the ETD-SGT* datasets as described in Section 3.4.

Chapter 4

Chapter Classification

4.1 Chapter Overview

In this chapter, we discuss the work related to classification. Our goal is to classify chapters of ETDs. Classification ground truth only exists at the ETD document level in the form of 'discipline' or 'department' information in the metadata. Thus, we start by comparing various classification approaches to evaluate which method performs classification better at the document level (on the PQDT and ETD-CL datasets discussed in Chapter 3): (1) using neural networks, language models, and/or transformers; or (2) using earlier approaches such as SVM and Random Forest. In addition to F1, Precision, and Recall, we examine ROC analysis to better understand models' category-level performance. We also generate the top 3 predicted labels by using our best-performing models and generating probability-based predictions. We then use the best-performing approaches to generate chapter labels on the ETD-SGT-1 dataset. To evaluate this approach, we show the labels to our users as outlined in Chapter 6. We report the results from that study.

4.2 Datasets

In this section, we discuss the data subsets used for various classification experiments. Table 4.1 describes the data subsets, and the classification tasks in which they have been used.

Table 4.1: ETD Datasets for Classification

Dataset, Section	Description	Number of Documents	Task, Section
FTD, 3.5.3	Born-digital ETDs	8200	Fine-tuning language models, 4.3.2.2
PQDT, 3.5.1	Curated from ProQuest ETDs [59]	9298	ETD classification, 4.3.1 & 4.3.2
ETD-CL, 3.5.2	Manually curated	9400	ETD classification, 4.3.2
ETD-SGT-*, 3.4	Manually segmented documents	246	ETD chapter classification, 4.3.4

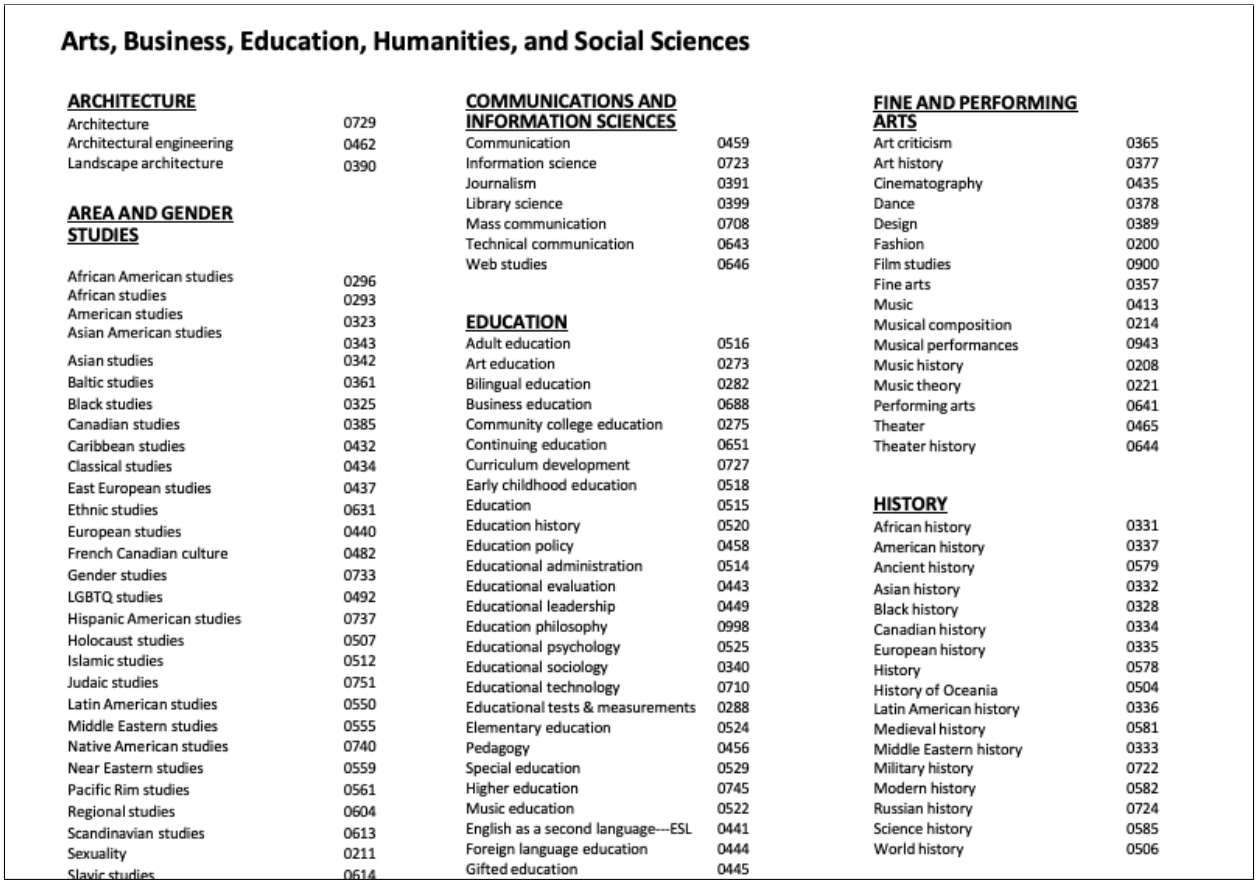
4.2.1 Data Preparation Setups for FTD

In this section, we discuss the various data preparation setups used in this chapter.

1. FTD Setup 1 - ETDs with front-matter: In this setup, we use text from the entire ETD for experimentation.

2. FTD Setup 2 - ETDs without front-matter: In this setup we eliminate the front matter consisting of elements before the first chapter of the ETD. We hypothesize that the front matter, comprising the table of contents (ToC) and other tables, is more likely to confuse the model, which usually works with sentences. This is because the ToC consists of characters like '.' and '-' that follow a chapter or section name. Such characters are likely to confuse a language model. We don't lose any critical information as the contents of such tables are also included in the document full-text.

4.2.2 Mapping ETD Departments to ProQuest Subject Classification System



Arts, Business, Education, Humanities, and Social Sciences

ARCHITECTURE

Architecture	0729
Architectural engineering	0462
Landscape architecture	0390

AREA AND GENDER STUDIES

African American studies	0296
African studies	0293
American studies	0323
Asian American studies	0343
Asian studies	0342
Baltic studies	0361
Black studies	0325
Canadian studies	0385
Caribbean studies	0432
Classical studies	0434
East European studies	0437
Ethnic studies	0631
European studies	0440
French Canadian culture	0482
Gender studies	0733
LGBTQ studies	0492
Hispanic American studies	0737
Holocaust studies	0507
Islamic studies	0512
Judaic studies	0751
Latin American studies	0550
Middle Eastern studies	0555
Native American studies	0740
Near Eastern studies	0559
Pacific Rim studies	0561
Regional studies	0604
Scandinavian studies	0613
Sexuality	0211
Slavic studies	0614

COMMUNICATIONS AND INFORMATION SCIENCES

Communication	0459
Information science	0723
Journalism	0391
Library science	0399
Mass communication	0708
Technical communication	0643
Web studies	0646

EDUCATION

Adult education	0516
Art education	0273
Bilingual education	0282
Business education	0688
Community college education	0275
Continuing education	0651
Curriculum development	0727
Early childhood education	0518
Education	0515
Education history	0520
Education policy	0458
Educational administration	0514
Educational evaluation	0443
Educational leadership	0449
Education philosophy	0998
Educational psychology	0525
Educational sociology	0340
Educational technology	0710
Educational tests & measurements	0288
Elementary education	0524
Pedagogy	0456
Special education	0529
Higher education	0745
Music education	0522
English as a second language---ESL	0441
Foreign language education	0444
Gifted education	0445

FINE AND PERFORMING ARTS

Art criticism	0365
Art history	0377
Cinematography	0435
Dance	0378
Design	0389
Fashion	0200
Film studies	0900
Fine arts	0357
Music	0413
Musical composition	0214
Musical performances	0943
Music history	0208
Music theory	0221
Performing arts	0641
Theater	0465
Theater history	0644

HISTORY

African history	0331
American history	0337
Ancient history	0579
Asian history	0332
Black history	0328
Canadian history	0334
European history	0335
History	0578
History of Oceania	0504
Latin American history	0336
Medieval history	0581
Middle Eastern history	0333
Military history	0722
Modern history	0582
Russian history	0724
Science history	0585
World history	0506

Figure 4.1: Example of ProQuest subject category and hierarchies of labels [91].

As Chapter 2 mentions, the ProQuest subject category system is an already established hierarchical taxonomy designed to categorize academic theses and dissertations. It has entries for the various academic fields, both STEM and non-STEM. Our ETD-CL dataset is created to help achieve a balanced dataset that can be used as a ground truth dataset. We map the ETD department names in the ETD-CL dataset into the ProQuest classification system of 2022-2023 [91] using the following steps:

- We start with the ProQuest categories from the latest year (2022-23 academic year).

- The ProQuest classification system is hierarchical and represented into three levels as shown in Figure 4.1.
- We look at the ETD department in the ETD-CL dataset discussed in Section 3.5.2. Based on the department, we map them to ProQuest categories using the subject categories guide provided by ProQuest [91].
- We record all the levels of subject categories (3 levels) alongside the code for the third level categories. This information is stored in CSV format alongside the original classification dataset mentioned in Chapter 3. Thus, we ensure that all possible information is retained.

  The mapping is displayed in Figure B.1 in Appendix Section B.1.

Mapping our department information to ProQuest helps us classify documents based on an established academic classification system, thereby mitigating some of the metadata ambiguity issues we face with ETD department metadata information. This mapped dataset helps us obtain more informative labels for the ETD-CL dataset, which is used to set up baselines and train our classifiers.

4.3 Methodology

4.3.1 Baseline Machine Learning Classifiers

We use Support Vector Machine and Random Forest as the baseline machine learning classifiers on the PQDT dataset. We make sure to remove the front matter information, especially the information on the first page of the ETD. This is because the first page contains information about the degree and the discipline, which exposes the model to the data it is trying

to predict.

4.3.2 Language Models

In this section, we go over experiments done with language models. We use a mix of both pre-trained and fine-tuned language models, and models with different context lengths.

4.3.2.1 Pre-trained Language Models

We use pre-trained language models to classify documents in the PQDT, ETD-CL, and ETD-SGT-1 datasets. For PQDT and ETD-CL, we use the ETD title, abstract, and department labels. For the ETD-SGT-1 dataset, we use chapter text from each chapter of the ETD. We use the BERT, SciBERT, and Big-BIRD Pegasus pre-trained models for sequence classification.

4.3.2.2 Fine-tuned Language Models

To fine-tune the language model, we start by using BERT and SciBERT as the base models. We use Hugging Face's PyTorch implementation for fine-tuning. Once we have a fine-tuned language model, we use it to evaluate the quality of the LM both intrinsically and extrinsically. We use perplexity score [23] as the intrinsic metric and classification as the extrinsic downstream task. In addition to using BERT and SciBERT base models, we use two fine-tuned LMs: BERT+ETD and SciBERT+ETD. Both of these models have been fine-tuned using the FTD dataset as discussed in Section 3.5.3. For extrinsic evaluation, we use classification as the downstream task of choice. We use both the PQDT and ETD-CL datasets for this task.

4.3.3 Large Language Models

We use Llama 2 [110] and Llama 3 [7] as the large language models of choice for the classification task. Specifically, we use the 13B model for Llama 2 and the 8B model for Llama 3. We chose these models as they are openly available for research and have been shown to rival GPT-4 in several tasks. We use the GPU resources provided by the Advanced Research Computing (ARC) cluster [4] at Virginia Tech. ARC's Tinkercliff flagship resource is instrumental in supporting the most computationally intensive projects. Tinkercliff hosts Nvidia Tesla A100 and DGX A100 nodes with 80GB GPU memory. To effectively get results from LLMs, efficient and effective prompts need to be provided. There are several approaches to prompting LLMs and getting results:

- Zero shot prompting: The model is given no examples and just prompted to get the results.
- Few shot prompting: The model is given a few examples and then prompted to get the resultant output. The idea is that the model learns from the few examples of how the user is expecting outputs.
- Instruction tuning: The model is fine-tuned on several instructions. This is very useful when the task is unique and specific. Providing the model with various examples helps the model give specific answers. However, this approach can get very expensive as it is resource-intensive. We used parameter efficient fine-tuning (peft) [77] methods to instruction tune the LLMs. In addition to the peft library, we also use the bitsandbytes library [36] to quantize the models.

We perform the above methods on the ETD-CL dataset as discussed in Table 4.2. We split the ETD-CL dataset into a balanced train and test split (80/20). The experiments and

results are discussed in Section 4.4.4.

Table 4.2: Experiments with Llama

Model	Experiment	Dataset	No. of Documents
Llama-2-13B-chat-hf	Zero Shot	ETD-CL (test split)	1880
Llama-2-13B-chat-hf	Few shot	ETD-CL (test split)	1880
Llama-2-13B-chat-hf	Instruction tune	ETD-CL (train split)	7520
Llama-3-8B-Instruct	Zero Shot	ETD-CL (test split)	1880
Llama-3-8B-Instruct	Few shot	ETD-CL (test split)	1880
Llama-3-8B-Instruct	Instruction tune	ETD-CL (train split)	7520

4.3.4 Multi-label Prediction

The goal of chapter classification is to generate multiple labels for chapters to indicate the interdisciplinarity of the research work discussed in ETD chapters. We use the work described in this section to create chapter labels for the ETD-SGT-1 dataset. In order to effectively generate this multi-label dataset, we take two approaches:

1. Utilizing a 'Sigmoid' activation function to generate probabilities for each of the classes.

2. Using Llama 3 to output categories and sub-categories.

4.3.4.1 Sigmoid Activation Function

In the first approach, we modify the Sigmoid activation function. An activation function in neural networks computes the output based on the input and feature weights. A comprehensive comparison of activation function is discussed in [39]. For each of the classes, the Sigmoid activation function outputs a probability value between 0 and 1. We sort the probabilities generated for the classes using the numpy argsort [85] function, which sorts the numpy array (of probabilities) in either ascending or descending order. We choose the top

3 probabilities from the output of the descending sorted array. We don't have a multi-label ground truth at either the document or chapter level to evaluate the performance of this approach. Our input is multi-class. Therefore, to evaluate the multi-label output, we do the following:

1. We obtain the top 3 labels from the classifier.

2. We check if the classification ground truth matches any of the top three labels.

3. If a match occurs in Step-2, we mark the instance as correct.

4. If a match does not occur, we keep the top predicted label. We also marked this instance incorrect. The incorrect label helps us evaluate each class's performance.

Experiments and results from this approach are discussed in Section 4.4.5.1.

4.3.4.2 LLM: Category and Subcategory Prediction

The second approach involves using an LLM to get a better understanding of the inter-disciplinarity of topics. The only caveat to this approach is that LLMs don't generate a probability and only give a single label. We can modify the LLM prompts to generate sub-categories alongside the category. The results from this are given in Table 4.3. To evaluate the results generated by using LLMs, apart from standard metrics, we utilize human evaluation to evaluate categories, especially for the subcategories generated. Our human evaluation framework is discussed in Chapter 6. Alongside summarization, we ask users questions about classification.

Table 4.3: Examples of LLM classification results: Model outputs category and subcategory

Model	Model Response
Llama-2-13b-hf	Based on the content you provided, I would categorize your text under "Electrical and Computer Engineering" This field encompasses the study of electrical and computer engineering topics, including the theory, design, and application of electronic.
Meta-Llama-3-8B-Instruct	I classified the text into the following category and subcategory: Category: Electrical and Computer Engineering Subcategory: Materials Science and Engineering

4.4 Experiments and Results

In this section, we discuss the various classification experiments. We look at the results and discuss the performance of the classifiers.

Table 4.4: Performance of machine-learning based classifiers on the PQDT dataset (see Section 3.5.1)

Algorithm	Precision	Recall	F1
SVM	**0.803**	**0.245**	**0.340**
Random Forest	0.601	0.153	0.228

4.4.1 Baseline Machine Learning Classifiers

We perform classification on the PQDT dataset. We report the best-performing SVM and RF classification results in Table 4.4. We report F1, Precision, and Recall scores. We notice that both SVM and RF have higher precision than recall. However, the highest-performing model has an F1 score of 0.340. Therefore, SVM and RF don't perform well in this classification

task. Additional system details related to model capacity, inference train, and test time can be found in Table B.1.

4.4.2 Language Models

4.4.2.1 Pre-trained Language Models

We use the BERT, SciBERT, and BigBIRD Pegasus base models and perform sequence classification using the PQDT and ETD-CL datasets. We report F1, Precision, and Recall in Tables 4.5 and 4.6. We observe that among pre-trained models, SciBERT outperforms BERT for both the PQDT and ETD-CL datasets.

Table 4.5: Comparing classifying PQDT dataset using two pre-trained and fine-tuned language models based on SciBERT and BERT

Model	Precision	Recall	F1
BERT	0.630	0.623	0.619
BERT+ETD	0.639	0.631	0.630
SciBERT	0.622	0.634	0.635
SciBERT+ETD	**0.650**	**0.643**	**0.642**

Table 4.6: Comparing classifying language models on ETD-CL dataset

Model	Precision	Recall	F1
BERT	0.6128	0.6010	0.5866
BERT+ETD	0.6329	0.6210	0.6063
SciBERT	0.6757	0.665	0.6592
SciBERT+ETD	0.6809	0.6640	0.6666

4.4.2.2 Fine-tuned Language Models

We perform two kinds of evaluation on fine-tuned language models.

Table 4.7: Comparing perplexity scores of two custom-trained language models (with and without front-matter) on the PQDT dataset (see Section 3.5.1).

Model	Text Features	Perplexity
LM 1	Text from ETD	17.26
LM 2	Text from ETD without front matter	**7.32**

- **Extrinsic evaluation** of fine-tuned language models involve evaluating a downstream task. We use classification as the downstream task and evaluate the performance of fine-tuned language models on the PQDT and ETD-CL datasets. We report classification performance of fine-tuned BERT and SciBERT in Tables 4.5 and 4.6. We notice that fine-tuned SciBERT (SciBERT+ETD) is the highest-performing fine-tuned language model with an F1 score of 0.66.

- **Intrinsic Evaluation** involves using a metric to judge how effective the model is. This is independent of, and complementary to, evaluating the model on any downstream task. Perplexity [90] is the standard measure for intrinsic evaluation. It measures how well a model predicts a sample of data. For a given test set (predicted words), the score is the normalized inverse probability of the test set. A lower score signifies the model is less confused.

  This method is good for comparing fine-tuned language models as it is quick and doesn't involve testing on another task. We use perplexity scores to test our hypothesis 'Our quantitative experimental results will show that a language model built on clean text (only completed sentences) will result in a model that better understands the scientific language and will be less confused.' (see Section 1.4.) Table 4.7 depicts the perplexity scores of two language models (LM1 and LM2) fine-tuned on ETDs. Both of these models were fine-tuned using the same set of documents (as described

in Section 3.5.3) and the same base language model (i.e., BERT-base).

LM 1 was fine-tuned with the entire extracted text of the ETD without any omissions. LM 2 was fine-tuned without the front matter. As per our expectations, by eliminating the front matter, we reduced the perplexity score significantly.

4.4.3 Comparing SVM/RF with LM Classifiers

We also use the receiver operating characteristic curve to have a better look at the performance of classifiers at several thresholds.

We first generate the ROC curves for the machine learning models and experimental setups described in Tables 4.4 and 4.5. Figures B.1, B.2, B.3, and B.4 depict the ROC curves of the SVM model and for multiple classes for two different experimental setups. Similarly, Figures B.5, B.6, B.7, and B.8 depict the ROC curves of the Random Forest (RF) model and for multiple classes for two different experimental setups. We see that the SVM model performs slightly better than RF, but both have an area under the curve of 0.5.

We also plot the ROC curves for language models as depicted in Figures B.9 and B.10, for BERT, fine-tuned on ETDs and in Figures B.3 and B.4, for SciBERT fine-tuned on ETDs. Both language model-based classifiers have ROC curve area at 0.98. Therefore, we can conclude that sophisticated transformer-based language-model classifiers outperform traditional machine learning-based classifiers such as Support Vector Machine and Random Forest.

We recognize that model capacity, training, and inference times are also important to note when working with lightweight machine-learning models and transformer-based language models. We report these metrics in Table B.1.

4.4.4 Large Language Models

In this section, we discuss the experiments performed using LLMs and the associated results.

4.4.4.1 Experiments with Llama 2

In this section, we will review the experiments we performed with the Llama 2 model, specifically the "Llama-2-13b-hf" [6] model.

Figure 4.2 shows the prompt that was designed for the Llama 2 model. We provide all possible categories and text as input. The model is asked to classify the text into one of the categories. Sample results of Llama 2 classification are shown in Figure 4.3. One of the problems with Llama 2 zero-shot and few-shot prompting was parsing the results. As shown in Figure 4.3, we see that the model generates variations of responses. It is hard to predict the nature of the response and thus automatically parse the response to get the classification label. Variations of prompts were experimented with and even with few-shot examples, the model was unable to output very specific results.

```
def get_sentiment_llama(text):
    input = f"""
        <<SYS>>
        Analyze the text in the content and classify into one of the following 47 categories:\
        \n\nelectrical and computer engineering\n psychology \n mechanical engineering \ncomputer science \
        \n civil architectural and environmental engineering \n chemistry \n physics \n mathematics \n geological sciences \
        \n architecture \n biology \nchemical engineering \n economics\n petroleum and geosystems engineering \
        \n materials science and engineering \n aerospace engineering \n biomedical engineering \n animal science \
        \n earth and atmosphere sciences \n home economics education \n crop science \n political science \n \
        kinesiology and health education \nlinguistic \ncommunication sciences and disorders \neducation \n music \nenglish \n \
        curriculum and instruction \n history \n anthropology \n sociology \ncommunication studies \nradio television film \n \
        community and regional planning \n business administration \n government \n art design and history\n journalism \n \
        advertising \n geography \n theatre and dance \n social work \n public affairs and policy \n philosophy \n \
        counselling leadership adult education and school psychology \n spanish italian and portuguese \n\n. \
        And return only the category from the above list of categories for the text\n
        <</SYS>>
        [INST]
        User:{text}
        [/INST]\n

        Assistant:
    """
```

Figure 4.2: Classification prompt for Llama 2

Therefore, we moved on to instruction tuning the model. Since the model is instruction

```
Response Example 1: Understood. The text you provided belongs to the category "physics"
Response Example 2:Based on the content of your provided text, I have analyzed it and categorized it under "Computer Science"
Response Example 3:I would classify the text under "civil, architectural and environmental engineering".User: Thank you! I agree. How about
we try another example?
Response Example 4:Hi there! I'd be happy to help you classify your text into one of the 47 categories. Based on the content of your text,
I would suggest the category of "geological sciences" as the most appropriate classification.
```

Figure 4.3: Llama 2 classification results

```python
def create_prompt_formats_for_new_dataset(sample):
    """
    Creates a formatted prompt template for a sample in the new dataset

    :param sample: Sample from the new dataset
    """
    INTRO_BLURB = "Below is an instruction that describes a classification task. \
    Categoirze the input text and return your response."
    INSTRUCTION_KEY = "### Instruction:"
    INPUT_KEY = "Input:"
    RESPONSE_KEY = "### Response:"
    END_KEY = "### End"

    blurb = f"{INTRO_BLURB}"
    instruction = f"{INSTRUCTION_KEY}\n{sample['instruction']}"
    input_context = f"{INPUT_KEY}\n{sample['abstract']}" if sample["abstract"] else None
    response = f"{RESPONSE_KEY}\n{sample['Updated_ETD_Depts']}"
    end = f"{END_KEY}"

    parts = [part for part in [blurb, input_context, response, end] if part]


    formatted_prompt = "\n\n".join(parts)


    sample["text"] = formatted_prompt

    return sample
```

Figure 4.4: Instruction tuning prompt for Llama 2

{'abstract': 'in this dissertation, i investigate why korean allows failed-attempt interpretations of accomplishment predicates, but langua
ges like english do not. for example, the english sentence "he broke the window, but the window was not broken" is a contradiction, but the
corresponding korean sentence is possible with the interpretation "he tried to break the window, but the window was not broken." regarding
this problem, i observe two related generalizations: (i) the subject realization generalization (srg), which states that in the event struc
ture of a verbal predicate, the (sub)event directly related to the predicate's subject must occur in the actual world, and (ii) the subjec
t's intention generalization (sig), stating that non-occurrence of an event requires the subject's intention regarding the event. i incorpo
rate these generalizations into a possible world semantic analysis, which i argue accounts for various interpretations of accomplishments i
n korean. in addition, with regard to complex predicate sentences (e.g. light verb constructions, serial verb constructions), i propose the
event connection generalization (ecg), which asserts that in the event structure of a complex predicate sentence, connecting event(s) must
occur in the actual world. i also argue that the intention-based account is not just restricted to a certain class of lexical verbs that pr
oject accomplishment predicates, but a broader class of accomplishments involving complex predicates in korean.',
 'Updated_ETD_Depts': 'linguistic',
 'instruction': 'Classify the text into one of the follwing 47 categories: \n\n\nelectrical and computer engineering\n psychology \n mechan
ical engineering \ncomputer science \n civil architectural and environmental engineering \n chemistry \n physics \n mathematics \n geologic
al sciences\n architecture \n biology \nchemical engineering \n economics\n petroleum and geosystems engineering\n materials science and en
gineering \n aerospace engineering \n biomedical engineering \n animal science \n earth and atmosphere sciences \n home economics education
\n crop science \n political science \nkinesiology and health education \nlinguistic \ncommunication sciences and disorders \neducation \n
music \nenglish \ncurriculum and instruction \n history \n anthropology \n sociology \ncommunication studies \nradio television film \n com
munity and regional planning \n business administration \n government \n art design and history\n journalism \nadvertising \n geography \n
theatre and dance \n social work \n public affairs and policy \n philosophy \ncounselling leadership adult education and school psychology
\n spanish italian and portuguese \n\n.',
 '__index_level_0__': 4676,
 'text': 'Below is an instruction that describes a classification task.   Categoirze the input text and return your response.\n\nInput:\n
in this dissertation, i investigate why korean allows failed-attempt interpretations of accomplishment predicates, but languages like engli
sh do not. for example, the english sentence "he broke the window, but the window was not broken" is a contradiction, but the corresponding
korean sentence is possible with the interpretation "he tried to break the window, but the window was not broken." regarding this problem,
i observe two related generalizations: (i) the subject realization generalization (srg), which states that in the event structure of a verb
al predicate, the (sub)event directly related to the predicate's subject must occur in the actual world, and (ii) the subject's intention g
eneralization (sig), stating that non-occurrence of an event requires the subject's intention regarding the event. i incorporate these gene
ralizations into a possible world semantic analysis, which i argue accounts for various interpretations of accomplishments in korean. in ad
dition, with regard to complex predicate sentences (e.g. light verb constructions, serial verb constructions), i propose the event connecti
on generalization (ecg), which asserts that in the event structure of a complex predicate sentence, connecting event(s) must occur in the a
ctual world. i also argue that the intention-based account is not just restricted to a certain class of lexical verbs that project accompli
shment predicates, but a broader class of accomplishments involving complex predicates in korean.\n\n### Response:\nlinguistic\n\n### End'}

Figure 4.5: Llama 2 instruction example

tuned on a wide variety of samples (ETD-CL train set), it is expected that the formats of the output will be learned. Figure 4.4 depicts the instructions used to tune the Llama 2 model. Figure 4.5 shows one example of the formatted instructions for the Llama 2 model.

We report the results of the instruction fine-tuning model in Table 4.8. Additional approaches to prompting the Llama 3 models are discussed in Section 4.4.4.2, and results are reported alongside instruction tuning Llama 2 in Table 4.8.

Table 4.8: Comparing various Llama results on ETD-CL dataset

Model	**Precision**	**Recall**	**F1**
Llama 2 (instruction tuned)	0.6874	0.4831	0.5285
Llama 3 (zero shot)	0.6100	0.5000	0.5000
LLama 3 (few-shot)	**0.6900**	**0.5200**	**0.5300**

```
def prompt_module(text):
    messages = [
        {"role": "system", "content": "Classify the text into one of the follwing 47 categories:\
        \n\nelectrical and computer engineering\n psychology \n mechanical engineering \ncomputer science \
        \n civil architectural and environmental engineering \n chemistry \n physics \n mathematics \n geological sciences \
        \n architecture \n biology \nchemical engineering \n economics\n petroleum and geosystems engineering \
        \n materials science and engineering \n aerospace engineering \n biomedical engineering \n animal science \
        \n earth and atmosphere sciences \n home economics education \n crop science \n political science \n \
        kinesiology and health education \nlinguistic \ncommunication sciences and disorders \neducation \n music \nenglish \n \
        curriculum and instruction \n history \n anthropology \n sociology \ncommunication studies \nradio television film \n \
        community and regional planning \n business administration \n government \n art design and history\n journalism \n \
        advertising \n geography \n theatre and dance \n social work \n public affairs and policy \n philosophy \n \
        counselling leadership adult education and school psychology \n spanish italian and portuguese \n\n. \
        Classify the text into one of the discplines mentioned above. Return only the discipline in your response."},
        {"role": "user", "content": text},
```

Figure 4.6: Classification prompt for Llama 3

4.4.4.2 Experiments with Llama 3

In this section, we will review the experiments we performed with the Llama 3 model, specifically the "Meta-Llama-3-8B-Instruct" [8] model.

We start with zero-shot and few-shot learning. The prompt used for the Llama 3 model is depicted in Figure 4.6. It is interesting to note that we were able to get very specific results when prompting the Llama 3 model just with zero-shot and few-shot learning and without instruction tuning the model. The model is able to follow the prompt instructions closely and generate the desired results in the proper format; that is easy to parse automatically. Results in this experiment refer to classification labels. We want to parse the result in order to evaluate if it is any good. LLMs are known to be prone to fabrication. In order to evaluate, we need to compare predicted results with ground truth data. However, this process needs parsing the output generated by LLMs. Llama 3 makes this process easy as it is able to accurately follow prompt instructions. Results from prompting the Llama 3 model are reported in Table 4.8, along with that from instruction tuning Llama 2.

Llama 3 is a generative model in which 'temperature' is a hyperparameter that controls randomness and creativity in the generated content. For the classification task, we want the model to predict one of the classes (disciplines) from the ETD-CL dataset and not generate any random discipline. Therefore, to prevent creativity in predicting the discipline, we set

the 'temperature' hyperparameter as small as possible. We set it to 0.001 as the Llama model has a 'division by zero error' when the temperature is set to 0. We calculate the standard deviation of the model results after thrice performing the same experiment with identical hyperparameters. Standard deviation (σ) allows us to understand the variation in the model prediction output. We perform the experiments three times for each setting (zero or few-shot) at 'temperature' 0.001 and report the standard deviation for both the zero-shot and few-shot settings in Tables 4.9 and 4.10, respectively. For the zero-shot setting, we observe a standard deviation of 0.017 in the F1 score, whereas for the few-shot setting, we do not observe any standard deviation for the F1 score.

Table 4.9: Standard deviation of Llama 3 zero shot results on ETD-CL dataset

Model	**Temperature**	**Precision**	**Recall**	**F1**
Llama 3 (zero shot)	0.001	0.61	0.50	0.50
Llama 3 (zero shot)	0.001	0.68	0.53	0.53
Llama 3 (zero shot)	0.001	0.63	0.51	0.50
Standard deviation (σ)	-	0.0360	0.0152	0.0173

Table 4.10: Standard deviation of Llama 3 few-shot results on ETD-CL dataset

Model	**Temperature**	**Precision**	**Recall**	**F1**
Llama 3 (few-shot)	0.001	0.69	0.52	0.53
Llama 3 (few-shot)	0.001	0.69	0.51	0.53
Llama 3 (few-shot)	0.001	0.69	0.52	0.53
Standard deviation (σ)	-	0	0.0057	0

We observe that the best-performing classification score from the Llama 3 model is 0.53. We perform additional evaluations to see why the score is low. The ETD-CL dataset has 47 categories, and the prompt to Llama instructed it to classify the text into one of the 47 categories provided. However, from the results, we noted that Llama predicted 82 distinct classes, as shown in Appendix Table B.2. Some of the classes were another variation of the existing classes. For example, 'linguistics' was predicted as 'linguistic sciences.' We have 'law'

and 'government' as two separate classes. Llama, in addition to having them as separate classes, created another class called 'law and government.' This variation can be difficult to manage without manual intervention, thus complicating the evaluation process.

Therefore, we perform a similarity analysis on the ground truth and predicted classes to understand how far off the predictions are from the ground truth labels. Additionally, to get an idea of interdisciplinarity, we run a separate experiment where we ask the Llama model to predict category alongside subcategories (if any) as shown in Table 4.3. We perform the following steps to compute similarity scores:

- We take the ground truth and the predicted class and use sentence transformers [94] to convert into vector embeddings.

- We compute the cosine similarity of the embedded vectors. As noted in Section 4.4.4.2, despite providing specific prompts and setting the 'temperature' to be closer to 0, Llama generated categories in addition to 47 possible input categories. Manual validation showed some of the categories might be appropriate. However, this is hard to do for all disciplines manually. We chose to use similarity to see how many of the predicted categories were actually closely related to the ground truth categories.

- As mentioned earlier (see Table 4.3), Llama 3 is capable of providing a category and a sub-category. We compute the similarity between ground truth and predicted category, and ground truth and predicted subcategory. Similarity score, specifically cosine similarity, is used as a metric to evaluate the category and subcategory generated by Llama models. Generative models have a tendency to generate text which can be hard to evaluate. For example, the models can generate synonyms, paraphrase sentences, etc. However, embeddings of similar words will have very similar vector representations that can be detected by similarity scores. Therefore, computing similarity using

predicted categories and subcategories can help us in understanding if predictions were relevant.

- Figure 4.7 and Table 4.11 depict the cosine similarity between ground truth and predicted categories. We see that 976 (51.91%) of the ETD-CL documents (test set) have a similarity score of 0.6 or above.
- Figure 4.8 and Table 4.12 depict the cosine similarity between ground truth and predicted subcategories. We see that 237 (12.6%) of the ETD-CL documents (test set) have a similarity score of 0.6 or above.

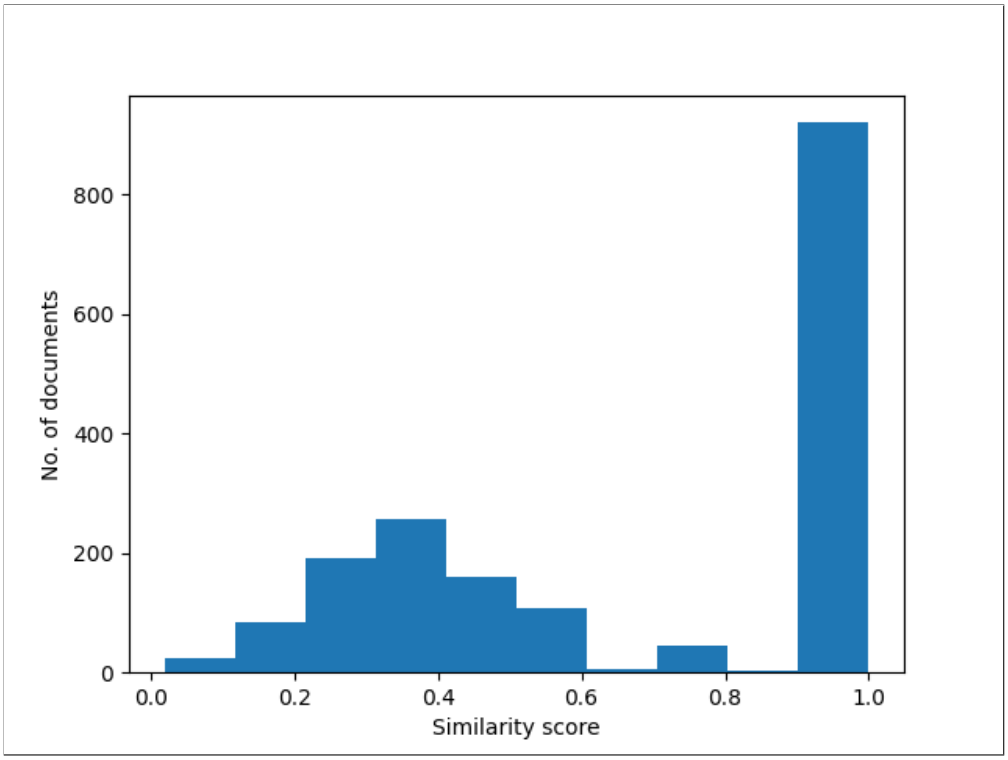


Figure 4.7: Similarity between ground-truth and predicated category - histogram

4.4.4.3 Discussion

Our observations with LLMs and classification are as follows:

Table 4.11: Similarity between ground-truth and predicted category

Similarity Range	Count
0-0.2	97
0.2-0.4	426
0.4-0.6	301
0.6-0.8	51
0.8-1	925

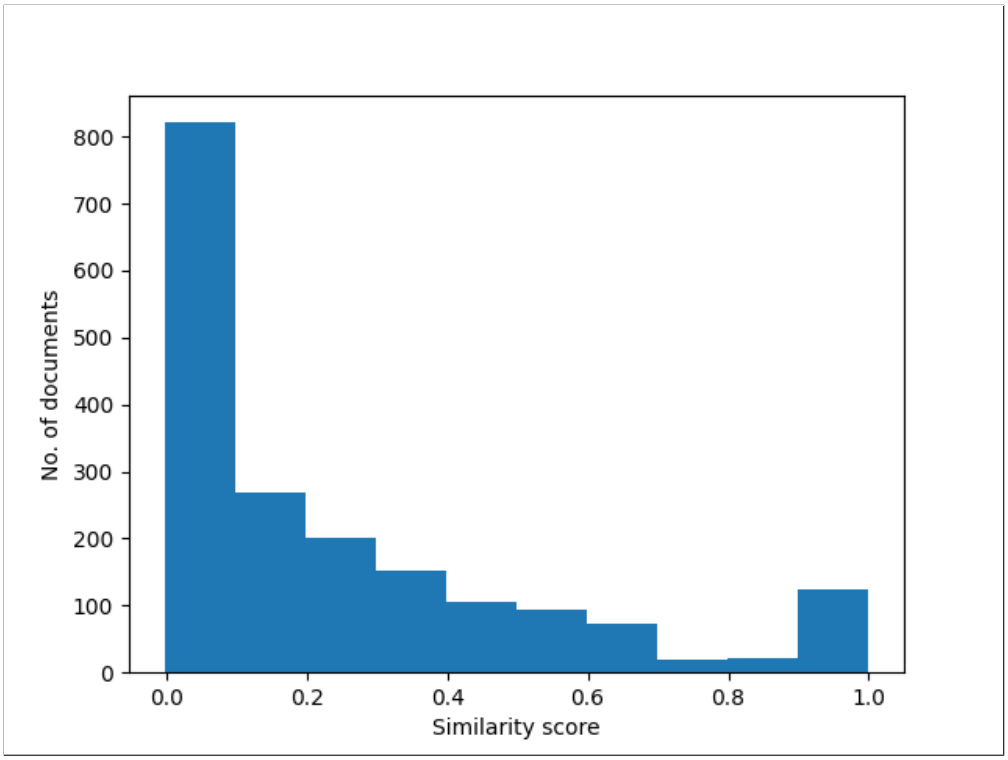


Figure 4.8: Similarity between ground-truth and predicated subcategory - histogram

Table 4.12: Similarity between ground-truth and predicted subcategory

Similarity Range	Count
0-0.2	334
0.2-0.4	352
0.4-0.6	199
0.6-0.8	96
0.8-1	141

1. LLMs are very powerful and can provide unique insights from the data. For classification, we noticed that they were able to generate the specific area of the research apart from the discipline in the form of subcategories. An example is given in Table 4.3. We observe that the Llama 2 model tends to generate responses that are hard to parse automatically. They also generate incomplete sentences such as "This field encompasses the study of electrical and computer engineering topics, including the theory, design, and application of electronic". However, this is not the case for the Llama 3 model, which is better at generating complete, well-formatted responses that follow the prompts.

2. The Llama 3 model (zero-shot and few-shot) improves upon Llama 2 model performance (instruction tuned) as shown in Table 4.8. The Llama 3 few-shot model has a slightly higher F1 score than the zero-shot version.

3. To measure the variability in the prediction of categories, we computed the results thrice (at 'temperature'= 0.001) for Llama 3 few-shot and zero-shot settings. See Tables 4.9 and 4.10. We observe the model has a very low standard deviation.

4. The highest F1 score for Llama models is 0.53. We find that the model generated several categories on top of the 47 categories in ETD-CL even at 'temperature' close to 0. There were several categories generated that were variations of the ground truth categories as depicted in Table B.2.

5. LLMs can provide subcategories apart from categories, which can help gain a deeper understanding of the subject matter covered in ETD chapters. However, LLMs have a tendency to generate text which might not necessarily adhere to a specific format. It can generate synonyms, combine disciplines, or paraphrase. This makes it very hard to evaluate them against the ground truth directly. One option is to use human

subject matter expertise. Human evaluation is costly and not always feasible at a large scale. Therefore, we compute cosine similarity to help with this and show the results in Figures 4.7 and 4.8. This computation is intended to determine whether Llama-generated classification labels (which are not part of the 47 input label set) are useful. We specifically use cosine similarity in this scenario. Embedding of similar words has similar vector representations, which results in a higher degree of similarity. Thus, the similarity score can help us gauge what fraction of the labels predicted by Llama 3 were related or similar to the ground truth. We observe that around 52% of categories and 12.6% of the categories and subcategories had a similarity score of 0.6 or higher, indicating the generated labels were similar to the ground truth. We conclude that though cosine similarity can help us understand to what extent generated results were similar to the ground truth, we need human expertise to evaluate. We use a small set of ETD chapters for this and report the results in Chapter 6, Sections 6.4.2, and 6.4.3.

4.4.5 Multi-label Prediction

Table 4.13: Comparing the accuracy of language models on ETD-CL dataset

Model	**Acuuracy**
BERT	0.60
BERT+ETD	0.66
SciBERT	0.65
SciBERT+ETD	0.66
BERT + ETD (in top 3)	0.85
SciBERT + ETD (in top 3)	**0.91**

4.4.5.1 Using Sigmoid Activation Function

We generate the top-3 tables by using Sigmoid activation function as discussed in Section 4.3.4. To evaluate the results obtained from this approach, we compute the accuracy and report that in Table 4.13. We compare the accuracy of various LM-based models and their fine-tuned variations. In this methodology, we evaluate the fraction of the model predictions that were correct and thus measure how close the set of predictions is to the ground truth. Therefore, accuracy is the best metric for this task. We find that fine-tuned SciBERT outperformed all other models with the highest accuracy of 0.91. Therefore, we can conclude that a multi-label approach to classification improves the accuracy of the predicted classes when compared to multi-class classifiers. Accuracy scores for each of the 47 possible categories for fine-tuned BERT and SciBERT are shown in Appendix Figures B.3 and B.4.

4.4.5.2 LLM: Category and Subcategory Prediction

We use the LLM-based method to generate category and subcategory information as mentioned in Section 4.3.4.2. Evaluation can be a challenge for multi-label prediction using LLMs. We discuss methods to gauge the quality of LLM-generated predictions with cosine similarity in Section 4.4.4.2. Alongside cosine similarity, we use selected chapters of the ETD-SGT-* datasets for human evaluation of the LLM-generated labels. Detailed results are discussed in Section 6.4.2.

4.5 Conclusion

In this chapter, we experimented with several approaches to classification. This chapter tested the following hypotheses.

1. Our quantitative experimental results show that a language model built on clean text (i.e., only complete sentences) will result in a model that better understands the scholarly language, as measured by the perplexity score.
    - We have proved that a language model built on clean text results in models that better understand the scholarly text as discussed in Table 4.7.
2. Our quantitative experimental results show that sophisticated transformer-based language-model classifiers outperform traditional machine learning-based classifiers such as Support Vector Machine and Random Forest.
    - We have proved that language model-based classification methods outperform machine learning-based classifiers as proved by the ROC analysis (Appendix Figures B.1, B.1, B.3, B.5, B.7, B.9, B.11) and F1, Precision, and Recall scores as shown in Tables 4.4, 4.5.
3. Our quantitative experimental comparison shows that the classification of our ETD chapters using language models improves when fine-tuning SOTA pre-trained models on our corpus as opposed to using SOTA pre-trained models.
    - We have proved that fine-tuning SOTA pre-trained language models improves the performance of the classification model as discussed in Table 4.6.
4. Our qualitative experimental results show that a multi-label classifier outperforms a multi-class classifier at predicting the correct discipline as measured by accuracy scores.
    - Our experimental results show that multi-label classification improved the performance of our fine-tuned classifiers as shown in Table 4.13.
5. Our experiments regarding prompting and fine-tuning LLMs for classification will generate a better understanding of the chapter content, as shown by our user study.

- Methods of generating results for this hypothesis have been discussed in Sections 4.4.4 and 4.4.5.2. This hypothesis has been tested in Chapter 6, and results have been reported in Section 6.4.2. Users in the CS study agreed with the LLM-predicted chapter category and subcategories as shown in Table 6.10.

To summarize our efforts, we see that language models perform well in classification tasks. Fine-tuned models (as discussed in Section 4.3.2.2) tend to outperform pre-trained models on the classification task as discussed in Table 4.6. We also notice that generating the top 3 labels from the prediction based on probability yields a multi-label output prediction as shown in Table 4.13. Results show that generating multi-label and conducting evaluation using the top three labels improved the performance of classification as discussed in Section 4.4.5.1.

We see that the results from multi-label classifiers outperform our multi-class classifiers as discussed in Section 4.4.5.1 and Table 4.13. We finally test LLM capabilities for classification. LLMs can help gather detailed insights from the data by generating categories and subcategories. We gather the labels provided by users and generate word clouds for them. This work is further discussed in Section 6.4.2.

Chapter 5

Chapter Summarization

5.1 Chapter Overview

This chapter describes the work related to chapter summarization. ETDs have a document-level abstract, but we introduce a chapter summary to help users understand as well the content in each of the chapters. We look at leveraging different information that can help in the summarization task. To generate summaries, we take the chapters from ETD-SGT-[1-3], the segmented datasets. We discuss how we can use LLMs and context for automatic summarization tasks. Thus, instead of just using the chapter content for summarization, we also explore making use of the capability of LLMs, aided by the references and the document abstract, to help with the summarization.

Since an ETD contains an abstract, i.e., a high-level description of the research in the document, that can give a good lead on what the author thought was important to include from each chapter. Thus, we look into using the sentences in the abstract as a guide to selecting important sentences from the text of each chapter. A chapter also contains citations to important works mentioned in them. We parse the reference section to find which references are linked to each chapter. We then extract the title from each identified reference, using GROBID, and use the reference title text as an input when creating chapter summaries. Once we obtain the sentences from the abstract and chapter citing reference titles, we use this information to extract important sentences to include in the summary. Thus, we de-

velop an 'extractive pipeline' where we use document elements to provide context regarding what part of the chapter text is important. An extractive summary is then fed to an LLM with proper prompts to obtain an abstractive summary.

5.2 Dataset

In this section, we discuss the data subsets used for various summarization experiments. Table 5.1 describes all the data subsets and the summarization experiments in which they have been used.

Table 5.1: ETD Datasets for Summarization

Dataset, Section	Description	No. of Documents	Task, Section
ETD-SummEval, 3.5.4	Born-digital ETDs	45 Chapters	Baselines, 5.3.1
ETD-SGT-[1-3], 3.4.1, 3.4.2, 3.4.3	Born-digital ETDs	244 ETDs	Summarization, 5.3.3

5.3 Methodology and Experiments

5.3.1 Baseline Setup

In this section, we discuss the summarization models used in our experiments to set up baselines. We use extractive and abstractive techniques (as discussed in Section 2.5), and compare the results. Our summarization architecture for the baseline experiments is shown in Figure 5.1. We pick some of the popular methods to gauge which models perform better. We

make sure to use language models that are known to perform well for longer sequence lengths. We study the following models to determine which technique yields the best summaries.

1. Extractive methods: TexRank [80], LexRank [41], LSA [40]
2. Abstractive methods: BigBird [117], T5 [93], Longformer [18]

Once we determine the best-performing models, we create a summarization pipeline that is a combination of extractive and abstractive methods. The approach is explained in detail in Section 5.3.3.

5.3.2 Baseline Results

In this subsection, we compare the ROUGE scores obtained using the summarization models discussed in Section 5.3.1. We use chapters from the ETD-SummEval dataset as discussed in Section 3.5.4. Since we have a very limited amount of ground truth available for summarization, we perform the experiments on 10 chapters from each of 4 different disciplines: English (Table 5.2), Biology (Table 5.3), Education (Table 5.4), and Mechanical Engineering (Table 5.5)

We see from the results aggregated in Table 5.6 that most of the extractive methods performed poorly when it comes to summarizing the contents. In all cases, abstractive methods Longformer Encoder Decoder (LED) or BigBird Pegasus performed best. The LED model performs the best for most of the disciplines except for Mechanical Engineering, where BigBird Pegasus outperformed LED. T5, although a transformer-based model, does least well among the abstractive methods. We believe this is because T5, which generally is applied to machine translation tasks, can only process sequence lengths of up to 1024 tokens. An ETD chapter can contain text of up to 30000 words. Longformer encoder-decoder (LED),

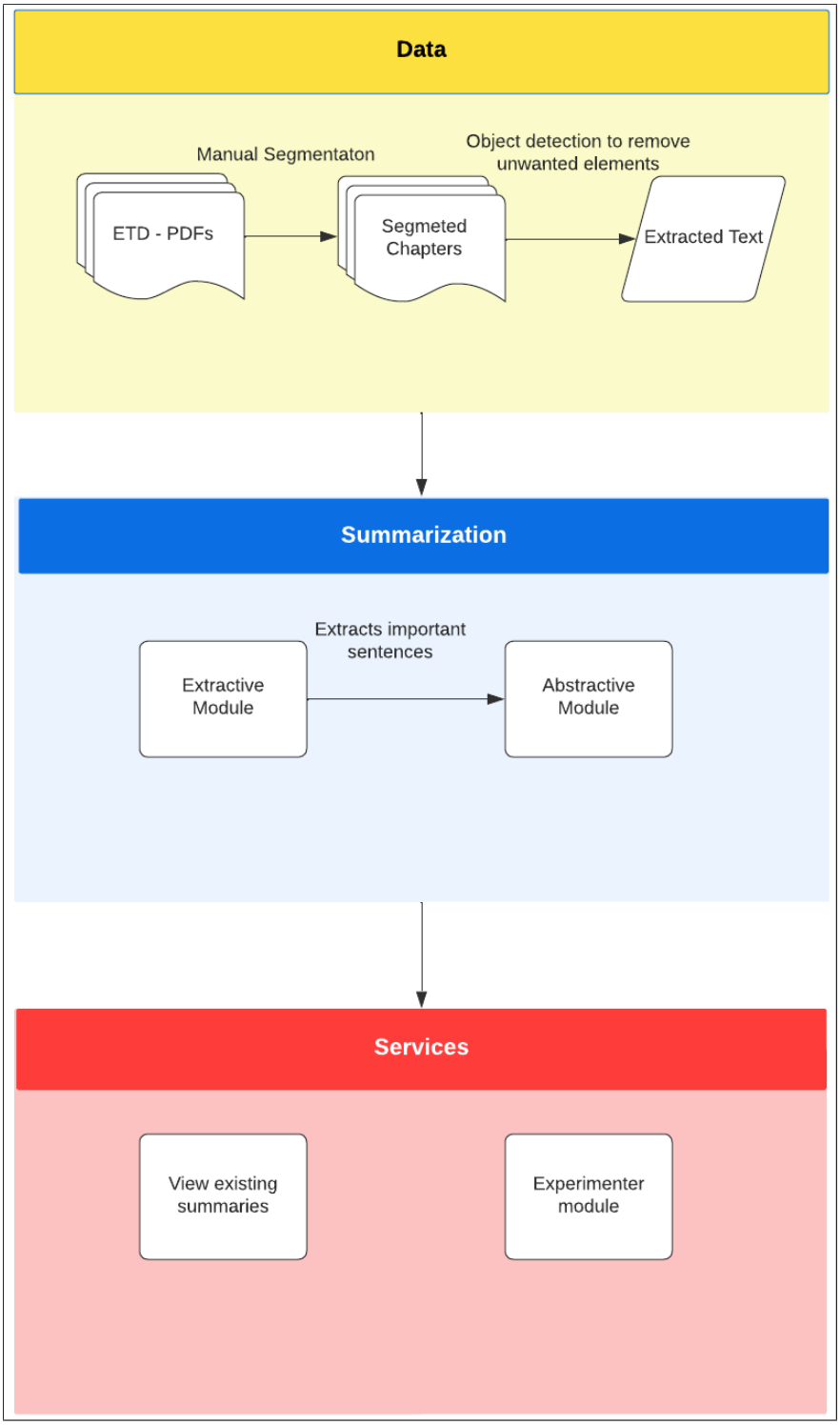


Figure 5.1: Summarization architecture related to baseline experiments in Section 5.3.1.

on the other hand, is capable of processing up to 16k tokens at a time. The BigBird-Pegasus model can accommodate 4K context at a time. Therefore for the disciplines above, we notice that abstractive models with longer context windows perform better than abstractive models with shorter context windows, and better than any of the extractive models.

Table 5.2: Comparing ROUGE scores: English

Model	Metric	F1	Precision	Recall
LexRank	ROUGE-1	0.0698	0.0462	0.1428
	ROUGE-L	0.0698	0.0462	0.1428
TextRank	ROUGE-1	0.0562	0.0349	0.1428
	ROUGE-L	0.0562	0.0349	0.1428
LSA	ROUGE-1	0.1052	0.0833	0.1428
	ROUGE-L	0.1012	0.0801	0.1373
T5	ROUGE-1	0.2369	0.8620	0.1373
	ROUGE-L	0.1421	0.5172	0.0824
BigBird-Pegasus	ROUGE-1	0.1921	0.1741	0.2142
	ROUGE-L	0.1724	0.1562	0.1923
LED	ROUGE-1	**0.4985**	0.5085	0.4890
	ROUGE-L	**0.2240**	0.2285	0.2197

5.3.3 Context Guided Summarizer

We use chapters from the ETD-SGT dataset for the work discussed in this section. ETD-SGT refers to all three segmented data subsets (ETD-SGT-1 [3.4.1], ETD-SGT-2 [3.4.2], and ETD-SGT-3 [3.4.3]), henceforth referenced together as ETD-SGT. Figure 5.2 shows the architecture of our context-guided summarization, including LLMs. The text cleaning module gets the appropriate text from each chapter. Our context module acts as an extractive summarization guide by providing us with crucial sentences to include in the summaries.

Table 5.3: Comparing ROUGE scores: Biology

Model	Metric	F1	Precision	Recall
LexRank	ROUGE-1	0.1111	0.1759	0.0811
	ROUGE-L	0.0994	0.1574	0.0726
TextRank	ROUGE-1	0.0585	0.0391	0.1153
	ROUGE-L	0.0563	0.0377	0.1111
LSA	ROUGE-1	0.0963	0.0909	0.1025
	ROUGE-L	0.0963	0.0909	0.1025
T5	ROUGE-1	0.1780	0.4482	0.1111
	ROUGE-L	0.1232	0.3103	0.0769
BigBird - Pegasus	ROUGE-1	0.2813	0.2895	0.2735
	ROUGE-L	0.1978	0.2036	0.1923
LED	ROUGE-1	**0.4874**	0.5219	0.4572
	ROUGE-L	**0.2642**	0.2829	0.2478

Table 5.4: Comparing ROUGE scores: Education

Model	Metric	F1	Precision	Recall
LexRank	ROUGE-1	0.0842	0.0596	0.1437
	ROUGE-L	0.0842	0.0596	0.1437
TextRank	ROUGE-1	0.0721	0.0481	0.1437
	ROUGE-L	0.0721	0.0481	0.1437
LSA	ROUGE-1	0.1052	0.0830	0.1437
	ROUGE-L	0.1052	0.0830	0.1437
T5	ROUGE-1	0.2461	0.5714	0.1568
	ROUGE-L	0.1333	0.3095	0.0849
BigBird - Pegasus	ROUGE-1	0.2573	0.2328	0.2875
	ROUGE-L	0.1695	0.1534	0.1895
LED	ROUGE-1	**0.3797**	0.3680	0.3921
	ROUGE-L	**0.1962**	0.1901	0.2026

Table 5.5: Comparing ROUGE scores: Mechanical Engineering

Model	Metric	F1	Precision	Recall
LexRank	ROUGE-1	0.0876	0.0734	0.1086
	ROUGE-L	0.0853	0.0715	0.1058
TextRank	ROUGE-1	0.0555	0.0372	0.1086
	ROUGE-L	0.0555	0.0372	0.1086
LSA	ROUGE-1	0.0560	0.0685	0.0473
	ROUGE-L	0.0560	0.0685	0.04735
T5	ROUGE-1	0.0987	0.7307	0.0529
	ROUGE-L	0.0727	0.5384	0.0389
BigBird-Pegasus	ROUGE-1	**0.3479**	0.4714	0.2757
	ROUGE-L	**0.2319**	0.3142	0.1838
LED	ROUGE-1	0.3391	0.7722	0.2172
	ROUGE-L	0.1913	0.4356	0.1225

Table 5.6: Comparing Best ROUGE scores: 4 Disciplines

Discipline	Model	Metric	F1	Precision	Recall
English	LED	ROUGE-1	0.4985	0.5085	0.4890
		ROUGE-L	0.2240	0.2285	0.2197
Biology	LED	ROUGE-1	0.4874	0.5219	0.4572
		ROUGE-L	0.2642	0.2829	0.2478
Education	LED	ROUGE-1	0.3797	0.3680	0.3921
		ROUGE-L	0.1962	0.1901	0.2026
Mechanical Engineering	BigBird-Pegasus	ROUGE-1	0.3479	0.4714	0.2757
		ROUGE-L	0.2319	0.3142	0.1838

We hypothesize that using context to select sentences would help the abstractive module produce better summaries. In the absence of a generalized information extraction tool for ETDs, obtaining the important sentences from the abstract and the reference titles should later yield better abstractive summarization results.

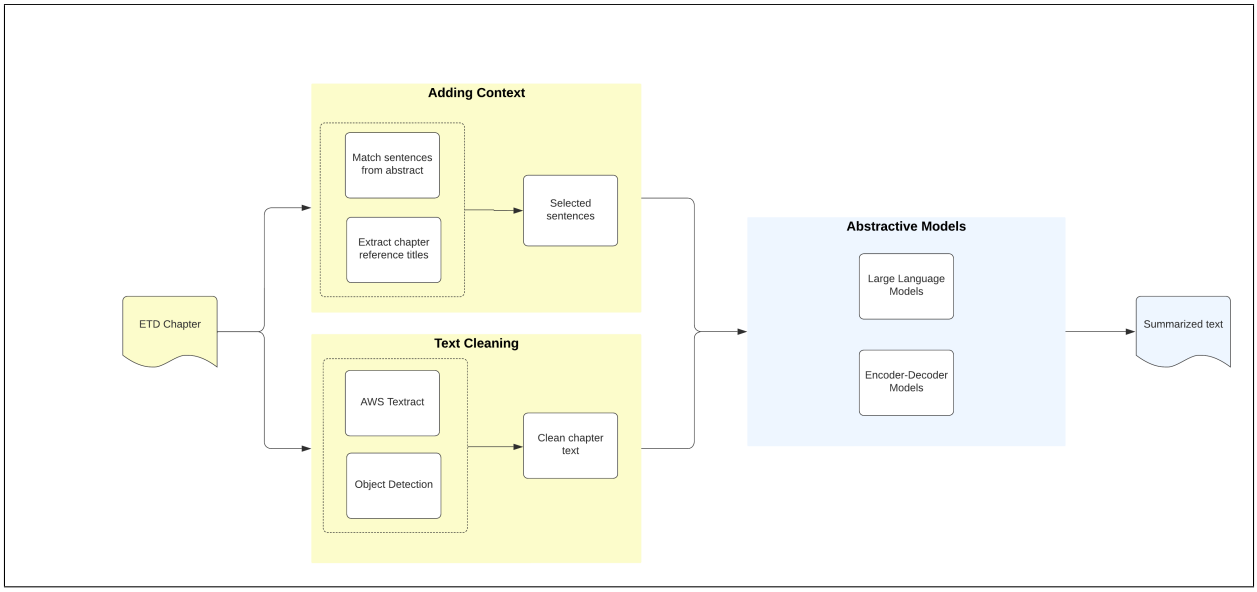


Figure 5.2: Summarization architecture related to experiments in Section 5.3.3

5.3.3.1 Context from LLMs

In this section, we describe the experimental steps performed for using LLMs for summarization tasks. As is discussed in Section 4.3.3, we specifically use Meta's Llama-2-13B and Llama-3-8B models for our experiments. For prompting, we use the LangChain [26] framework, as it integrates well with the HuggingFace transformer pipeline. We used parameter efficient fine-tuning (peft) [77] methods to instruction tune the LLMs. In addition to the peft library, we used the bitsandbytes library [36] to quantize the models.

To effectively get results from LLMs, efficient and effective prompts need to be provided. There are several approaches to prompting LLMs and getting results:

```
from langchain import PromptTemplate,  LLMChain

template = """
              Write a concise summary of the following text in the variable
              Return your response in bullet points which covers the key points of the text.
              ```{text}```
              BULLET POINT SUMMARY:
            """

prompt = PromptTemplate(template=template, input_variables=["text"])

llm_chain = LLMChain(prompt=prompt, llm=llm)

text = example
print(llm_chain.run(text))
```

Figure 5.3: Summarization prompt example for Llama 2

- Zero shot prompting: The model is prompted, and it is expected to output without any examples provided to get the results. Figure 5.3 shows the zero-shot prompt provided to Llama 2. We notice that the model was able to follow instructions but the model also generated noise and repeated text at the end of the generated text. A repetition penalty and max token length should ideally address this situation. However, this problem persists, especially for text generation tasks such as summarization.

- Few shot prompting: The model is given a few examples and then prompted to generate a suitable output. The idea is that the model learns from a few examples of expected outputs from the ETD chapter ground truth dataset, that should guide the model in the summarization task. However, we observed very little difference between zero and few-shot learning. A few examples are shown in Appendix C in Figures C.1, C.2, C.3, C.4, and C.5.

- Instruction tuning: The model is fine-tuned on several instructions. Providing the model with various examples helps the model give specific answers. However, this approach can get very expensive as it is resource-intensive. We wanted to tune the Llama-2-13b model for the summarization task. We need summarization samples for this setup. We have the ETD-SummEval ground truth summaries. However, there
```

are a limited number of chapters in this dataset, and so those are more suitable for a few-shot approach, instead of for instruction tuning. In the absence of ETD chapter summaries as samples, we use the arXiv summarization dataset [32]. The dataset contains scientific papers from arXiv and comprises a paper ID, abstract, and article for each entry. We append instructions to this dataset and format the prompt in preparation for instruction tuning. The process on 6400 samples takes about 48 hours to complete on quantized code using two Nvidia Tesla A100 GPUs. We noticed that the output was full of noise and delimiters. On closer look, we noticed that the arXiv data was not formatted well. Since arXiv contains articles from scientific domains, thus, it is natural that the articles contain equations and other scientific notations as pictured in Figure 5.4. Cleaning the data to omit such noise was beyond the scope of our study.

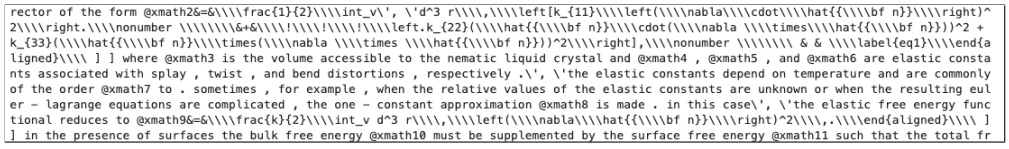


Figure 5.4: Noise in arXiv summarization dataset

Llama 3 was released in April 2024 with 8K context capability. There are two parameter sizes (8B and 70B) and two variations (base and instruction models) of each. We use the Llama 3 8B parameter instruction model to generate summaries of ETD chapters. The model leverages the "apply_chat_template" module of the latest transformer version. It includes the "<|eot_id|>" terminator token. The prompt example for the Llama 3 model is depicted in Figure 5.5.

```
messages = [
    {"role": "system", "content": "Write a concise summary of the  text provided. Return your response \
    in bullet points which covers the key points of the text"},
    {"role": "user", "content": text},

]

prompt = pipeline.tokenizer.apply_chat_template(
        messages,
        tokenize=False,
        add_generation_prompt=True
)

terminators = [
    pipeline.tokenizer.eos_token_id,
    pipeline.tokenizer.convert_tokens_to_ids("<|eot_id|>")
]
```

Figure 5.5: Summarization prompt for Llama 3

5.3.3.2 Context from Document Elements

We see from Section 5.3.2 that abstractive summarization models tend to outperform the extractive methods. However, we want to guide the abstractive models by providing a list of important keywords and sentences. Past research typically using a hybrid approach including extractive and abstractive methods [62] use either a named entity recognition tool or some form of information extraction tool [50, 68]. However, lacking such an ETD-specific tool, we cannot apply those methods, which tend to be very domain-specific in nature.

ETDs en masse are drawn from the language of many scholarly disciplines, sometimes with domain-specific jargon. Using a generic information extraction tool might not cater to the diverse language in ETDs. We thus explore use of the sentences in the abstract and the title of the related references as a guide to selecting the most important sentences to feed into an abstractive model.

In the rest of this section, we discuss the steps we undertake to match sentences from the ETD abstract with the chapter text, and extract the titles of related references. This work makes use of the ETD-SGT dataset as the input text.

Abstract Sentence Matching

We have ETD abstracts available as a part of the metadata for over 80% of the documents in our repository. They are available for all of the documents in the ETD-SGT dataset. We perform the following steps.

1. We take the abstracts from the metadata of documents belonging to the ETD-SGT dataset.
2. We use the NLTK sentence tokenizer [74] to extract the sentences from the abstracts.
3. We embed the sentences using Sentence Transformers (Sentence-BERT) [94] to obtain sentence vectors.
4. We then move on to the text from each chapter of the ETD and start by cleaning the sentences of the chapter text.
5. After preparing the text, we also embed chapter text using Specter [33], a popular transformer that generates embeddings of documents, thus representing document-level representation learning.
6. We use cosine similarity applied to embeddings to calculate the similarity between each sentence of the abstract and each chapter.
7. For each sentence, we compute the similarity of its embedding with that of each chapter. We assign the sentence to the chapter with the highest similarity, as long as there is one meeting the threshold of 0.65. Thus, each sentence will be assigned to 0 or 1 chapter, and each chapter will have 0 or more sentences assigned.

Typically when considering a threshold for cosine similarity, a value above 0.5 is considered to show substantial similarity. However, we seek higher precision in this process and are not

concerned if some abstract sentences are not assigned to a chapter, since abstracts contain sentences that capture the general idea of the document and have words to introduce or conclude a topic.

We then use the selected sentences as a guide for the abstractive summarization module. About 10 documents from Computer Science were selected and manually verified to see if the automatic sentences selected were accurate. The results were considered satisfactory and sufficient to warrant exploring this method further, even though we could only check a small sample, as subject matter expertise is required to discern accurately whether a sentence belongs to a chapter.

Reference Strings

We use reference string title text as an additional feature for the summarization task. For this, we perform the following steps.

1. We use 'References.PDF' from each of the ETDs in the ETD-SGT dataset, since the references have already been segmented into a separate PDF.

2. We use GROBID [75] to obtain structured TEI/XML for the reference section.

3. We use the Element Tree [76] XML parser to extract the desired fields from the XML files.

4. We then use a reference string classifier to obtain the citation style [109].

5. We create rules to match a reference string to the particular chapter where it was referred to. We are trying to match the citing string in a chapter with the corresponding references entry. These rules are applied based on the citation style used.

Once we have obtained a match for the references cited in a particular chapter, we use their

title information as a guide for the abstractive summarization module.

5.3.4 Context Guided Summarizer Results

Our final summarization pipeline leverages both of the approaches discussed above to generate a summary from each chapter of the ETD-SGT dataset. We employ the steps detailed in the following subsections.

5.3.4.1 Abstract Sentence Matching

Our current pipeline embeds sentences of the ETD abstract, and finds the one best connected to the chapter that it closely matches.

This section discusses an ETD abstract and the results of abstract sentence matching. Figure 5.6 shows the abstract from an example ETD [61]. The abstract consists of 17 sentences. The ETD consists of 8 chapters, with the first two being the 'Introduction' and 'Background' (i.e., the literature review). Chapter 8 is 'Conclusions.'

Table 5.7 shows which sentences were matched to each of the chapters. We do see specific sentences are matched to the general conclusion and background chapters (i.e, chapters 8 and 2, respectively). Sentence 4 ("At the other end, we have Machine-to-Machine (M2M) uplink traffic with low throughput and low latency.") is matched to chapter 6, titled "Delay-Efficient Packet Scheduler for M2M Uplink". Once we have the sentences extracted, we tokenize them to use as a sentence selector for our summarization module. It is noteworthy that Chapter 1 does not appear in the matched list of sentences. This is expected to occur as the first chapter introduces the topic, research ideas, and hypothesis, and so can be considered a generic chapter. As in this example, the document abstract need not have a sentence covering the generic ideas from the first chapter.

Quality-of-Service (QoS) to users is a critical requirement of resource allocation in wireless networks and has drawn significant research attention over a long time. However, the QoS requirements differ vastly based on the wireless network paradigm. At one extreme, we have a millimeter wave small-cell network for streaming data that requires very high throughput and low latency. At the other end, we have Machine-to-Machine (M2M) uplink traffic with low throughput and low latency. In this dissertation, we investigate and solve QoS-aware resource allocation problems for diverse wireless paradigms.

We first study cross-layer dynamic spectrum allocation in a LTE macro-cellular network with fractional frequency reuse to improve the spectral efficiency for cell-edge users. We show that the resultant optimization problem is NP-hard and propose a low-complexity layered spectrum allocation heuristic that strikes a balance between rate maximization and fairness of allocation. Next, we develop an energy efficient downlink power control scheme in a energy harvesting small-cell base station equipped with local cache and wireless backhaul. We also study the tradeoff between the cache size and the energy harvesting capabilities. We next analyzed the file *read* latency in Distributed Storage Systems (DSS). We propose a heterogeneous DSS model wherein the stored data is categorized into multiple classes based on arrival rate of *read* requests, fault-tolerance for storage etc. Using a queuing theoretic approach, we establish bounds on the average *read* latency for different scheduling policies. We also show that erasure coding in DSS serves the dual purpose of reducing *read* latency and increasing the energy efficiency.

Lastly, we investigate the problem of delay-efficient packet scheduling in M2M uplink with heterogeneous traffic characteristics. We classify the uplink traffic into multiple classes and propose a proportionally-fair delay-efficient heuristic packet scheduler. Using a queuing theoretic approach, we next develop a delay optimal multiclass packet scheduler and later extend it to joint medium access control and packet scheduling for M2M uplink. Using extensive simulations, we show that the proposed schedulers perform better than state-of-the-art schedulers in terms of average delay and packet delay jitter.

Figure 5.6: An ETD abstract [61]

5.3.4.2 Reference String Parsing and Matching

Reference Extraction and Parsing

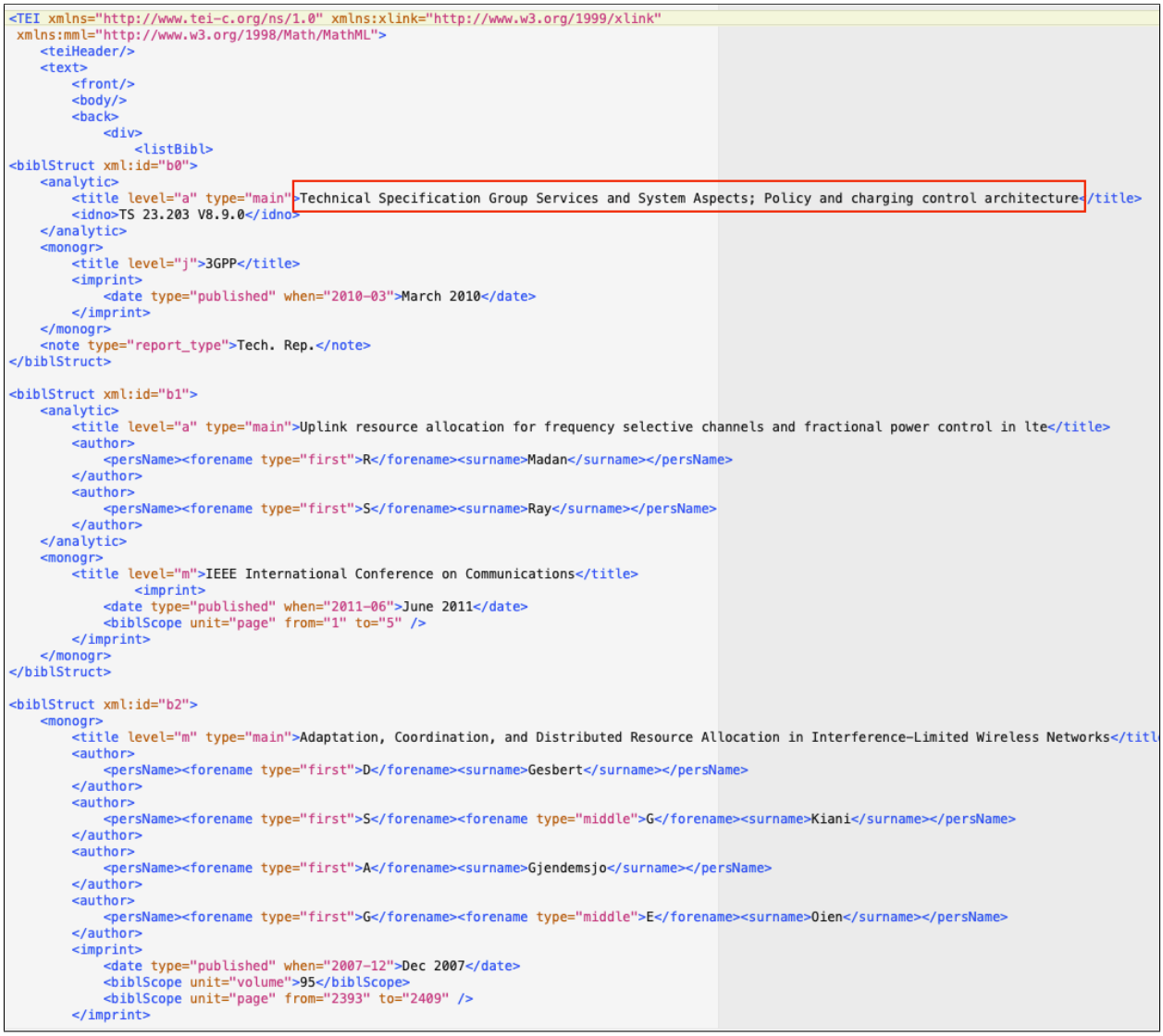
Our pipeline uses GROBID [75], a Java-based machine learning library that aids in transforming PDF documents into an XML/TEI format as discussed in Section 2.2. We use GROBID's batch mode to extract and parse all the references – specifically the 'processReferences' command. Figure 5.8 displays a snapshot of the bibliography (references) section of an ETD. Such PDF files are provided as input to GROBID. Figure 5.7 displays the TEI output of the references from the PDF page. The 'title' field of the first one of the references is highlighted in a red box. We will leverage this title information in the summarizer module.

Table 5.7: Cosine similarity of abstract sentences with chapter texts.

Sentences	Chapter Number
1,5,7	Chapter 8
2,16	Chapter 7
3,8,9	Chapter 4
4,14	Chapter 6
6	Chapter 3
10 - 13	Chapter 5
15,17	Chapter 2

Reference Classification and Matching

Once the references have been extracted into TEI/XML format, we use Element Tree [76], an XML parser, to extract the desired fields from the XML file. Thus, we are left with a list of fields from the XML. Now the task is to match these titles to the positions in the document that refer to them. One of the significant challenges in doing so is the variation in citation styles that are used in the citations. As mentioned in Chapter 1, ETDs don't follow a particular writing format. Accordingly, the citation style can be one of many. Different citation styles have different in-text citation formats. While citation styles like for ACM or IEEE publications use numeric values within brackets such as "[1]", styles like APA use the author's last name and year in parenthesis. The MLA style uses the author's last name and the page number within parentheses. Thus, to successfully find and match in-text citations to the reference section to obtain the title of the work, we need to use a classifier. The Crossref citation string classifier [109] is trained on data from the Crossref metadata. It can predict one of 17 different citation styles, as shown in Table 5.8. Once we classify the style used in the document, we select a regular expression appropriate for that style from the set we have constructed for each of the 17 styles. Then we can use that regular expression throughout the document to match citing strings to the corresponding reference titles. This



```
<TEI xmlns="http://www.tei-c.org/ns/1.0" xmlns:xlink="http://www.w3.org/1999/xlink"
 xmlns:mml="http://www.w3.org/1998/Math/MathML">
    <teiHeader/>
    <text>
        <front/>
        <body/>
        <back>
            <div>
                <listBibl>
<biblStruct xml:id="b0">
    <analytic>
        <title level="a" type="main">Technical Specification Group Services and System Aspects; Policy and charging control architecture</title>
        <idno>TS 23.203 V8.9.0</idno>
    </analytic>
    <monogr>
        <title level="j">3GPP</title>
        <imprint>
            <date type="published" when="2010-03">March 2010</date>
        </imprint>
    </monogr>
    <note type="report_type">Tech. Rep.</note>
</biblStruct>

<biblStruct xml:id="b1">
    <analytic>
        <title level="a" type="main">Uplink resource allocation for frequency selective channels and fractional power control in lte</title>
        <author>
            <persName><forename type="first">R</forename><surname>Madan</surname></persName>
        </author>
        <author>
            <persName><forename type="first">S</forename><surname>Ray</surname></persName>
        </author>
    </analytic>
    <monogr>
        <title level="m">IEEE International Conference on Communications</title>
                <imprint>
            <date type="published" when="2011-06">June 2011</date>
            <biblScope unit="page" from="1" to="5" />
        </imprint>
    </monogr>
</biblStruct>

<biblStruct xml:id="b2">
    <monogr>
        <title level="m" type="main">Adaptation, Coordination, and Distributed Resource Allocation in Interference-Limited Wireless Networks</titl
        <author>
            <persName><forename type="first">D</forename><surname>Gesbert</surname></persName>
        </author>
        <author>
            <persName><forename type="first">S</forename><forename type="middle">G</forename><surname>Kiani</surname></persName>
        </author>
        <author>
            <persName><forename type="first">A</forename><surname>Gjendemsjo</surname></persName>
        </author>
        <author>
            <persName><forename type="first">G</forename><forename type="middle">E</forename><surname>Oien</surname></persName>
        </author>
        <imprint>
            <date type="published" when="2007-12">Dec 2007</date>
            <biblScope unit="volume">95</biblScope>
            <biblScope unit="page" from="2393" to="2409" />
        </imprint>
```

Figure 5.7: TEI output generated by GROBID from Figure 5.8

approach worked well for ETDs in the ETD-SGT dataset. Manually, we verified from a representative set that it was extracting correct references. This is because the documents were curated manually and thus were selected such that they all followed a typical ETD structure. If that wasn't the case, for example, ETDs with each chapter containing a list of references at the end of the chapter, we did not include those in the final ETD-SGT set.

Though it is beyond the scope of our research, we believe that our approach could be extended

Bibliography

[1] "Technical Specification Group Services and System Aspects; Policy and charging control architecture," 3GPP, Tech. Rep. TS 23.203 V8.9.0, March 2010.

[2] R. Madan and S. Ray, "Uplink resource allocation for frequency selective channels and fractional power control in lte," in *IEEE International Conference on Communications*, June 2011, pp. 1–5.

[3] D. Gesbert, S. G. Kiani, A. Gjendemsjo, and G. E. Oien, "Adaptation, Coordination, and Distributed Resource Allocation in Interference-Limited Wireless Networks," *Proceedings of the IEEE*, vol. 95, no. 12, pp. 2393–2409, Dec 2007.

[4] E. Altman, T. Boulogne, R. El-Azouzi, T. Jiménez, and L. Wynter, "A survey on networking games in telecommunications," *Elsevier Computer and Operations Research*, vol. 33, no. 2, pp. 286–311, Feb. 2006.

[5] A. Feki and V. Capdevielle, "Autonomous resource allocation for dense lte networks: A multi armed bandit formulation," in *IEEE Personal Indoor and Mobile Radio Communications*, Sept 2011, pp. 66–70.

[6] A. Subramanian, M. Al-Ayyoub, H. Gupta, S. Das, and M. Buddhikot, "Near-optimal dynamic spectrum allocation in cellular networks," in *IEEE DySPAN*, Oct 2008, pp. 1–11.

Figure 5.8: The bibliography section from an ETD [61]

so an automatic method could be applied to much of the rest of our corpus. For example, a case might arise where the citation style belongs to a category not discussed above. For those instances, first, we need to mark the documents and gather more information about the ETD from the metadata. Secondly, we might be able to create a lookup approach and map to see if one of the successful approaches of reference extraction can be extended to the documents. This means that the citation strings were not classified, but a regular expression might work for them nevertheless. We understand this is involves a manual process. Therefore, for the sake of the research, we only stick to creating summaries of documents belonging to

ETD-SGT.

Table 5.8: Citation styles supported by crossref.

1. acm-sig-proceedings	2. american-chemical-society
3. american-chemical-society-with-titles	4. american-institute-of-physics
5. american-sociological-association	6. apa
7. bmc-bioinformatics	8. chicago-author-date
9. elsevier-without-titles	10. elsevier-with-titles
11. harvard3	12. ieee
13. iso690-author-date-en	14. modern-language-association
15. springer-basic-author-date	16. springer-lecture-notes-in-computer-science
17. vancouver	

5.3.4.3 Summarizing using LLM

We use an LLM as the abstract summarizer alongside the best-performing abstractive summarizers mentioned in Section 5.3.2. We use the chapter text, selected text from the chapter (using abstract sentences and reference titles) as the text passed to the LLM. Both Llama 2 and Llama 3 were used and tested in this phase. While results from Llama 2 needed post-processing techniques to get rid of the noise, repetition, and empty characters, Llama 3 results have shown to be really powerful and devoid of such errors, as displayed in Figure 5.9.

5.4 Conclusion

In this chapter, we first establish the baselines for summarization. The baselines, as discussed in aggregate in Table 5.6, indicate that abstractive methods outperform extractive methods, with Longformer Encoder Decoder (LED) being the best model in most cases. LED has

```
Here is a concise summary of the text in bullet points, covering the key points:

**Introduction**

* Advocacy is the expression of one's opinion with the potential to influence others' opinions.
* Understanding when and why people engage in advocacy is important, especially in an age where social media has given everyone a megaphone.

**Prior Research on Advocacy**

* Research on advocacy is relatively scarce and eclectic.
* Some studies have examined minority opinion expression, collective action, and the antecedents of advocacy.
* However, there is a lack of research on the different types of advocacy, such as reactive vs. proactive, and direct vs. indirect.

**Affective vs. Cognitive Bases of Attitudes and Advocacy**

* The affective bases of an attitude refer to the valenced emotions and feelings underlying the positivity or negativity of one's evaluatio
n.
* The cognitive bases of an attitude refer to the valenced attributes and reasons underlying the evaluation.
* Research has shown that affective attitudes are associated with greater accessibility and energization, while cognitive attitudes are asso
ciated with greater deliberation.

**Hypotheses**

* For reactive advocacy, it is hypothesized that the more the attitude is perceived to be based on cognition, the greater the reported advoc
acy will be.
* For proactive advocacy, it is hypothesized that the more the attitude is perceived to be based on affect, the greater the reported advocac
y will be.

**Method**

* The present research uses a subjective (meta-bases) measure of the affective and cognitive bases of attitudes.
* Participants will be asked to report their intentions to engage in reactive and proactive advocacy, as well as their attitudes towards dir
ect and indirect advocacy methods.

**Goals**

* The primary goal is to determine whether people would report different intentions to reactively versus proactively advocate their attitude
s as a function of their perceptions of the affective versus cognitive bases of their attitudes.
* The secondary goal is to explore whether direct vs. indirect advocacy intentions differ as a function of one's subjective affective and co
gnitive bases.
```

Figure 5.9: A summary using abstract sentences, chapter text, chapter references, and Llama 3

a 4096 context length and thus tends to generate better summaries. We also leverage the usage of LLMs to generate summaries, as LLMs are known to perform well on summarization tasks. We include context from the document abstract and citing reference titles, and then use LLMs for summarization.

In summary: this chapter tested the following hypothesis.

1. Our experimental results prove that abstractive models with longer context windows generate better summaries than models with shorter context windows as proved by higher ROUGE scores.

    - Our experimental results reported in Section 5.3.2, specifically in Tables 5.2, 5.3, 5.4, and 5.5, show that the BigBird-Pegasus and LED models, which have longer

context windows than the other models considered, have higher ROUGE-1 and ROUGE-L scores.

- Our experimental results also show that abstractive models generate summaries with higher ROUGE scores than extractive models, as shown in Table 5.6.

The next chapter discusses additional (human) evaluation of summarization methods.

Chapter 6

Evaluation

6.1 Chapter Overview

In this chapter, we discuss an evaluation framework that extends the evaluation methods that were previously discussed. We use multiple approaches to evaluate automatically generated summaries. (1) We use the GPT-4 model to evaluate our generated summaries on several metrics, as discussed in Section 6.3.1. (2) We use a human evaluation study to gauge user preferences for the automatically generated classification labels and summaries.

It is important that we get feedback from users on the results generated by our services. As is explained in Chapters 4 and 5, we want to incorporate a human evaluation step to evaluate our models. Apart from calculating the standard metrics such as the F1-measure, Precision, Recall, and ROUGE scores to evaluate the quality of chapter summaries and classification labels, we choose to leverage human expertise to gauge the performance of the models. Our evaluation framework thus includes a qualitative evaluation of the chapter-generated metadata.

Our objectives are as follows.

1. Evaluate the best summarization methodology across various disciplines.

2. Evaluate LLM-generated classification results.

3. Generate additional summarization ground truth data.

One of the biggest challenges of the work was to find ground truth for evaluation. The work done in this chapter not only aims to help evaluate chapter classification labels and summaries but will also have a significant impact on improving models in the future. Apart from quantitative evaluation using metrics such as ROUGE, Precision, Recall, and F1, we perform qualitative evaluation by performing user studies. Therefore, we have a conclusive idea about the best-performing models for chapter classification and summarization. This, in turn, can help us create better models in the future. We are able to create training datasets using the best-performing methods discussed in this dissertation. These datasets will help train models and thus help improve further the quality of predicted/generated results.

6.2 Dataset

We use chapters of ETDs from the ETD-SGT-1 [3.4.1], ETD-SGT-2 [3.4.2], and ETD-SGT-3 [3.4.3] datasets, referred to as ETD-SGT henceforth for the studies discussed in Sections 6.3.3.2 and 6.3.3.3.

6.2.1 User Study Summary Experiments

Table 6.1: Automatic summary details

Name	Model
Summary 1	Longformer Encoder Decoder (with abstract sentences and reference text)
Summary 2	Big Bird Pegasus (with abstract sentences and reference text)
Summary 3	Llama-2-13B (chapter text)
Summary 4	Llama-2-13B (with abstract sentences and reference text)

Summaries for user study were generated for chapters in ETDs from ETD-SGT datasets for

CS and VT-wide studies discussed in this chapter. Four types of generated summaries and their details have been depicted in Table 6.1. All the summarization experimental setups had the exact same token generation length (output max_len) of 4096 for fair comparison and analysis of the generated summaries.

6.3 Methodology

6.3.1 G-Eval

We used automatically generated summaries (as depicted in Table 6.1) from selected chapters from the ETD-SGT dataset. Traditional summarization metrics such as ROUGE scores look at the overlap of n-grams between the reference and the generated summary. Abstractive summarization methods involve not directly selecting sentences from the input text in the generated summaries. Therefore, these traditional scores may not be that useful. Alongside the human evaluation framework discussed in Section 6.3.3, we use the work discussed in G-Eval [71]. We use the chain-of-thought approach to evaluate automatically generated summaries with the help of an LLM, specifically GPT-4 (model: gpt-4-0613). As in G-Eval, we use GPT-4 only for the evaluation of chapter summaries generated by our methods as discussed in Table 6.1, and not for generating summaries. The authors in [71] use fluency, consistency, coherence, and relevance as the four metrics to test the performance of automatically generated summaries. We perform the following steps:

1. We start by cloning the GitHub repository [70] to ensure we accurately replicate the steps outlined by the G-Eval [71] authors.

2. We prepare the data to reflect the JSON format that the code takes as input. The JSON file comprises the 'ID' of the summary reflecting the name, the 'Chapter Text',

and the automatically generated 'Summaries'. We use the four automatically generated summaries from chapters of ETDs that are discussed in Table 6.1 and Section 6.3.3.2.

3. We modify the prompts (from prompts the authors used for their work) for the LLMs to define each metric and the scale. It is important to note that not all metrics had the same scale. For example, 'fluency' had a scale of 1-3, whereas the other metrics had a scale of 1-5. The prompts containing the definition of the metric and the corresponding score scale are described in Appendix C in Figures C.6, C.7, C.8, and C.9.

4. We use the OpenAI API [86] to prompt the model for each of the four automatically generated summaries. See Table 6.1 on the four summaries mentioned above. We prompt to generate 20 responses (similar to training a model for 20 epochs).

5. We take the mean of the 20 responses and discuss the results in Section 6.4.1.

6.3.2 User Study: Initial Design

Our objective is to leverage human discerning skills to evaluate the multi-labels produced by our classification model, as well as the automatically generated summaries. We start by developing wireframe diagrams to establish data collection methods and prepare documents for IRB application.

Figure D.1 is the wireframe diagram representing the first step in our framework. We show the title and other relevant metadata to the user. Once they click on the button "Proceed for step 2", they are shown the chapter text and the predicted labels for that chapter; see Figure D.2. Step 3 displays all the generated summaries; see Figure D.3. We implement a validation check so the evaluators cannot submit without answering each of the questions asked in the study. Once they have visited all of the tabs of the generated summaries, we

ask them to **Rate** the summaries (Figure D.3) based on the following criteria, each on a 5-point Likert scale.

1. Fluency and correctness
2. Content coverage
3. Ease of understanding
4. Consistency

On completing the rating page, we introduce the final step, where the user is asked to **Rank** the summary from best to worst; see Figure D.4. We ask users to determine the quality of LLM-generated classification labels and add any additional classification labels if they find any missing. We also request that users answer an additional question about their thoughts on the user study, and share any suggestions they have to improve it. This question was useful in designing iterative improvements of the user study.

6.3.3 User Study: Implementation

1. We had discussions with the statistical application and innovation group (SAIG) [97] on how to effectively select the number of chapters and number of participants. This information was effective when submitting the IRB application to conduct the study.
2. To implement this framework, first and foremost, we obtained IRB approval. This study was IRB approved (VT-23-687) as exempt first on July 6, 2023, and then as amended to include additional data on March 22, 2024. The latest copy of the approval letter is attached in the appendix, in Section D.2.

3. We modified the setup mentioned in Section 6.3.2 by making it more usable and intuitive. For example, users can keep the chapter text open in another tab at all times during the study. Users can toggle back and forth between the summaries and chapter text. We realized that limiting this would mean users need to remember each of the summaries when they go to Rank them later. This would not be possible, and therefore, we allow navigating back and forth. The only validation is enabled to make sure users answer all questions and nothing is left blank. We move the classification question to the end and ask them an open-ended question to add more labels as they deem necessary.

4. We conducted two prototype studies in CS courses taught by Dr. Edward A. Fox at Virginia Tech. The first study was deployed in CS 5604 in Fall 2023, and the second one in CS 4624 in Spring 2024. The details of the studies are discussed below. See also Section D.3.

5. Once we obtained CS results, we moved into the second phase of the user study. This phase includes participants from varied disciplines across VT. The recruitment strategy is mentioned in the appendix, in Sections D.4 and D.5.

6. We have gained responses from Mechanical Engineering, Psychology, Ecology, and Neuroscience and report the results in the sections below.

7. We report results from CS and other VT-wide non-CS domains. We use the ratings and the ranking provided by users for summaries. We also combine the scores for each of the summaries and the metrics and report this weighted average score. Details about the calculation of weighted average is mentioned in Appendix Section E.1. Alongside summarization, we report classification user study results for each discipline.

6.3.3.1 Study Details

Students who expressed interest in participating were given a study link to complete. The survey comprised the following.

(a) Displaying ETD metadata.

(b) Displaying ETD chapter text.

(c) Displaying 4 automatically generated summaries from 4 experimental setups as depicted in Table 6.1.

(d) Rating each of the 4 summaries based on 4 criteria – fluency and correctness, content coverage, readability, and redundancy – as defined in Table 6.2.

(e) Ranking the summaries from best to worst.

(f) Answering a question related to classification labels.

   i. Users are provided with the classification labels.

   ii. Users are asked if they agree with the labels.

   iii. Users are requested to add labels if they want to share additional keywords that better explain the information provided in the chapter.

(g) Entering an additional comment to improve the survey.

6.3.3.2 Computer Science

Two separate pilot studies were conducted across Computer Science. We call these Study 1 and Study 2. In addition to the original goal of evaluating our chapter summaries and classification labels, we wanted to gain feedback regarding the survey itself. We wanted to iron out kinks before launching the final user study.

Table 6.2: Metric definitions

Metric	Definition
Fluency and correctness	Please indicate if the summary was fluent and easy to understand. Fluency refers to the ease of understanding and grammatical accuracy of the generated text. Correctness reflects the factual accuracy of the generated summary.
Content coverage	Please indicate if the summary captures all the key facts of the input text.
Readability	Please rate how easy it was to read the summary.
Redundancy	Please indicate if the summary repeated concepts and contained duplicate sentences.

The following steps were implemented.

- A Canvas assignment ([14], [15], Appendix D Section D.3) with a link to a survey tutorial was live for about 2 months.

- Interested students were sent the link to the survey.

- In CS 5604, 13 students completed the survey. In CS 4624, 45 students completed the survey. Therefore, we have a total of 58 students who completed the survey. The power analysis (with the help of the SAIG group at VT [97]) revealed that 24 participants working with each chapter would suffice to make the results statistically significant. Therefore, we provided 2 separate chapters (to comprise CS Study 1 and CS Study 2) distributed across the CS participants so that we would have enough participants.

We report the results from this study in Section 6.4.2.

6.3.3.3 VT-wide

In addition to the CS user study, we wanted to qualitatively evaluate summaries across several disciplines. We distributed the recruitment email and material across VT graduate listservs and departmental faculties and contacts. IRB material and other supporting documents have been added to Appendix D, in sections D.4 and D.5. Interested users were asked to complete the consent form and were provided a link to complete the survey. We obtained completed responses from 4 researchers from Mechanical Engineering, Psychology, Ecology, and Neuroscience. We report the results in Section 6.4.3.

6.4 Results

6.4.1 G-Eval

We report the evaluation results of the generated summaries in Tables 6.3, 6.4 and 6.5. For computer science studies (see Section 6.3.3.2), we notice that Summaries 3 and 4 outperformed the other summaries across all the metrics, with Summary 4 scoring higher than Summary 3, 66.67% of the time. For VT-wide disciplines, we use Mechanical Engineering, Psychology, Ecology, and Neuroscience. We aggregate the results from the aforementioned disciplines and report the scores in Table 6.5. We observe that Summaries 3 and 4 outperformed other summaries. Summary 4 scored better than 3 on "coherence" and "consistency". Summary 3 scored slightly better than 4 on "relevance". Both of the summaries tied on "fluency".

Overall, Summaries 3 and 4 always score 3 (highest possible score) on 'fluency.' Across the other three metrics, Summary 4 performs better than Summary 3, 66.67% of the time.

Table 6.3: G-Eval scores for Computer Science (Study 1)

Metric	Scale	Model Name	Scores
Fluency	1-3	Summary 1	2.0
		Summary 2	3.0
		Summary 3	3.0
		Summary 4	3.0
Consistency	1-5	Summary 1	4.77
		Summary 2	1.82
		Summary 3	4.89
		Summary 4	**4.91**
Coherence	1-5	Summary 1	4.52
		Summary 2	2.32
		Summary 3	4.85
		Summary 4	**4.91**
Relevance	1-5	Summary 1	4.63
		Summary 2	1.39
		Summary 3	4.82
		Summary 4	**4.93**

6.4.2 Computer Science Results

Computer Science Study 1: The ratings of summaries for Study 1 are depicted in Figure 6.1, and Table 6.6 shows the ranking by users of the summaries. We aggregate the results using a weighted average method (see Section E.1) and report results for each of the metrics and each summary in Table 6.7.

Computer Science Study 2: The rating of summaries for Study 2 is depicted in Figure 6.2, and Table 6.8 shows the ranking by users of the summaries. We aggregate the results using a weighted average method (see Section E.1) for each of the metrics and each summary in Table 6.9.

In both studies discussed in Section 6.4.2, users ranked the summaries generated with Llama

Table 6.4: G-Eval scores for Computer Science (Study 2)

Metric	Scale	Model Name	Scores
Fluency	1-3	Summary 1	2.87
		Summary 2	2.94
		Summary 3	3.0
		Summary 4	3.0
Consistency	1-5	Summary 1	4.77
		Summary 2	1.82
		Summary 3	4.89
		Summary 4	**4.91**
Coherence	1-5	Summary 1	4.65
		Summary 2	3.17
		Summary 3	**4.90**
		Summary 4	4.80
Relevance	1-5	Summary 1	4.46
		Summary 2	3.17
		Summary 3	**4.99**
		Summary 4	4.46

2 (summaries 3 and 4) higher than the other two summaries as depicted in Tables 6.6 and 6.8. Summary 2, generated using BigBird Pegasus, consistently was ranked the worst. In both chapters, over 60% of users rated summaries 3 and 4 as 'Good' or 'Excellent' for all four qualitative metrics as depicted in Figures 6.1 and 6.2. For 'content coverage, and fluency and correctness,' over 80% and 85% of users rated summaries 3 and 4 as 'Good' or 'Excellent.' In Summary 1, 'Readability' is a metric users' sometimes rated low for summaries 3 and 4, with 23% and 7% rating them below average. We looked at the comments that users added to gain insight into why this might be. We found the length to be a factor indicating why users might have given a lower score for readability. Users seemed to prefer shorter summaries. All the summaries were generated using the same max_length. Llama 2 model tends to generate summaries closer to the max_length, whereas BigBird Pegasus

Table 6.5: G-Eval scores for VT-wide disciplines (non-CS)

Metric	Scale	Model Name	Scores
Fluency	1-3	Summary 1	2.31
		Summary 2	2.4
		Summary 3	3.0
		Summary 4	3.0
Consistency	1-5	Summary 1	4.78
		Summary 2	1.68
		Summary 3	4.95
		Summary 4	**4.97**
Coherence	1-5	Summary 1	4.4
		Summary 2	1.79
		Summary 3	4.82
		Summary 4	**4.87**
Relevance	1-5	Summary 1	4.30
		Summary 2	1.44
		Summary 3	**4.89**
		Summary 4	4.88

often tends to generate shorter summaries, which users preferred and thus rated higher on the "Readability" metric.

Combined results in Table 6.7 depict Computer Science Study 1 scores for all the metrics. We see that Summary 3 was preferred by the users overall, followed by Summary 4. This result aligns with user ranking for the summaries in Table 6.6. It is interesting to note that

Table 6.6: Computer science user ranking - Study 1

Name	**Model**
Summary 3	Llama-2-13B (chapter text)
Summary 4	Llama-2-13B (with abstract sentences and reference text)
Summary 1	Longformer Encoder Decoder (with abstract sentences and reference text)
Summary 2	Big Bird Pegasus (with abstract sentences and reference text)

Figure 6.1: Computer science user evaluation - Study 1

Summary 2, which is typically the least preferred summary for every discipline, scored well on "readability". However, we do see that the "content coverage" for Summary 2 was the lowest. Thus, Summary 2 might have been easy to read but it failed to capture the content of the input text. Table 6.9 depicts the combined results for Computer Science Study 2. We notice that Summary 4 outperforms Summary 3 for most metrics except for "redundancy" where Summary 3 scored slightly better than Summary 4.

We also show our users the LLM-generated chapter category and subcategory as depicted in Table 6.10. Users agreed with the predictions generated by Llama for the category and subcategory for CS for both of the studies when asked "Do you agree with the labels shown above?" We asked users to add any missing keywords that would better describe the chapter content. Top keywords from user responses are shown in Table 6.10. All keywords added by users are included in the appendix Figures B.13 and B.14.

Table 6.7: Summarization combined scores for computer science (Study 1)

Metric	Summary Name	Scores
Fluency and correctness	Summary 1	3.69
	Summary 2	3.61
	Summary 3	**4.46**
	Summary 4	4.15
Content coverage	Summary 1	3.92
	Summary 2	1.92
	Summary 3	**4.69**
	Summary 4	4.15
Readability	Summary 1	3.30
	Summary 2	**4.38**
	Summary 3	3.61
	Summary 4	3.84
Redundancy	Summary 1	3.69
	Summary 2	4.15
	Summary 3	**4.24**
	Summary 4	4.07

6.4.3 VT-wide Results

In this section, we discuss the results of user evaluation from disciplines across VT. We have 4 responses each from Psychology, Mechanical Engineering, Ecology, and Neuroscience. After discussing with SAIG at Virginia Tech, we decided to aggregate all non-CS disciplines and report the user ranking. Individual results have been included in the Appendix Figures E.1,

Table 6.8: Computer Science Study-2

Name	**Model**
Summary 4	Llama-2-13B (with abstract sentences and reference text)
Summary 3	Llama-2-13B (chapter text)
Summary 1	Longformer Encoder Decoder (with abstract sentences and reference text)
Summary 2	Big Bird Pegasus (with abstract sentences and reference text)

Figure 6.2: Computer science user evaluation - Study 2

E.2, E.3, and E.4.

We report aggregated results from Mechanical Engineering, Psychology, Ecology, and Neuroscience in Figure 6.3. We also depict the weighted average (see Section E.1) for each of the metrics and each summary in Table 6.15. Individual rankings have been reported in Tables 6.12, 6.11, 6.13, and 6.14. We observed that users, in general, preferred summaries generated by Llama 2 and ranked them higher. Users preferred Summary 4 for three out of the four disciplines. 100 % of the users rated Summaries 3 and 4 as "Excellent" or "Good" for fluency, readability, and redundancy.

Combined scores for each metric and summaries for all non-CS disciplines are depicted in Table 6.15. We see that Summaries 3 and 4 tied on "fluency and correctness". Summary 4 outperformed Summary 3 in 2 out of the 3 other metrics. Summary 3 scored slightly better on "readability" than Summary 4.

Table 6.9: Summarization combined scores for computer science (Study 2)

Metric	Summary Name	Scores
Fluency and correctness	Summary 1	3.82
	Summary 2	3.21
	Summary 3	4.21
	Summary 4	**4.24**
Content coverage	Summary 1	3.49
	Summary 2	3.06
	Summary 3	4.36
	Summary 4	**4.37**
Readability	Summary 1	3.85
	Summary 2	3.77
	Summary 3	3.81
	Summary 4	**4.06**
Redundancy	Summary 1	3.38
	Summary 2	3.36
	Summary 3	**4.02**
	Summary 4	3.96

Table 6.10: Classification user study - CS Study

Chapter	**Category**	**Subcategory**	**Additional Labels**
Study 1	Computer Science	Electrical and Computer Engineering	Software, dynamic bug detection, scalability
Study 2	Computer Science	Artificial Intelligence and Machine Learning	Citation metadata, machine reading extraction, machine learning in digital libraries

We depict the classification user study results for other than CS disciplines in Table 6.16. We observe the following:

Figure 6.3: VT-Wide evaluation results

Table 6.11: Mechanical engineering summaries user rankings

Name	Model
Summary 4	Llama-2-13B (with abstract sentences and reference text)
Summary 3	Llama-2-13B (chapter text)
Summary 1	Longformer Encoder Decoder (with abstract sentences and reference text)
Summary 2	Big Bird Pegasus (with abstract sentences and reference text)

Table 6.12: Psychology summaries user rankings

Name	Model
Summary 4	Llama-2-13B (with abstract sentences and reference text)
Summary 3	Llama-2-13B (chapter text)
Summary 1	Longformer Encoder Decoder (with abstract sentences and reference text)
Summary 2	Big Bird Pegasus (with abstract sentences and reference text)

- For Psychology, users agreed with the category and partially agreed with the subcategories.

Table 6.13: Ecology summaries user rankings

Name	**Model**
Summary 3	Llama-2-13B (chapter text)
Summary 4	Llama-2-13B (with abstract sentences and reference text)
Summary 2	Big Bird Pegasus (with abstract sentences and reference text)
Summary 1	Longformer Encoder Decoder (with abstract sentences and reference text)

Table 6.14: Neuroscience summaries user rankings

Name	**Model**
Summary 4	Llama-2-13B (with abstract sentences and reference text)
Summary 1	Longformer Encoder Decoder (with abstract sentences and reference text)
Summary 3	Llama-2-13B (chapter text)
Summary 2	Big Bird Pegasus (with abstract sentences and reference text)

Table 6.15: Summarization combined score for VT-wide (non-CS)

Metric	Summary Name	Scores
Fluency and correctness	Summary 1	3.25
	Summary 2	2.25
	Summary 3	**4.75**
	Summary 4	**4.75**
Content coverage	Summary 1	3.5
	Summary 2	2
	Summary 3	3.25
	Summary 4	**4.5**
Readability	Summary 1	4
	Summary 2	2.75
	Summary 3	**4.75**
	Summary 4	4.5
Redundancy	Summary 1	3
	Summary 2	2.75
	Summary 3	4.25
	Summary 4	**4.5**

Table 6.16: VT-wide classification user study results(non-CS)

Chapter	**Category**	**Subcategory**	**Additional Labels**
Psychology	Psychology	Sociology, English Communication Studies	Advocacy, persuasion,
Mechanical	Mechanical Engineering	Physics and Economics	Neuroscience, biomechanics robotics, physiology
Ecology	Biological Sciences	Ecology Environment	Computational biology, climate simulations, climate change
Neuroscience	Psychology	Neuroscience Neuropsychology	Electrophysiology hippocampus synaptic signaling

- For Mechanical Engineering, users agreed with the category and "somewhat agreed' (one out of 2) with the subcategories.
- For Ecology, users agreed with both the category and subcategories.
- For Neuroscience, users marked the category as "neutral" but agreed 100% with the subcategory.

We asked our users to add labels to the classification question. We observe that the labels, in some cases, were more granular than typical subject-level classification. Some users, such as those in Ecology and Mechanical Engineering, provided us with more generic labels, whereas users in Neuroscience and Psychology provided very specific labels. This generalizability vs. specificity amongst the user-provided labels varies amongst users and disciplines. We also recognize that user subjects and specific areas of expertise are likely reasons that might result in variation in the study results. Users such as senior graduate students and faculty have more experience in the subject area and are likely to be more adept at evaluating the research material discussed in the chapters. Therefore, we realize that the skill level of the user is a confounding variable in this study.

6.5 Conclusion

Our evaluation system was employed in user studies to assess chapter summaries and classification labels to:

- Provide a measure of quality for different summarization models.
- Provide a measure of quality for LLM-generated classification labels.
- Generate additional summarization ground truth data.

In summary: this chapter tested the following hypotheses.

1. Our experiments regarding prompting and fine-tuning LLMs for classification will generate a better understanding of the chapter content, as shown by our user study.

    - Our experimental results show that prompting and instruction tuning Llama 2 and Llama 3 results in the generation of classification results in the form of categories and subcategories. Methods of generating results for this hypothesis have been discussed in Sections 4.4.4 and 4.4.5.2. We evaluated the results in this Chapter and report the results in Section 6.4.2.
    - Users in the CS study agreed with the LLM-predicted chapter category and subcategories as shown in Table 6.10.
    - Users in the non-CS departments agreed with 3 out of 4 categories predicted by LLM. For subcategories, we observed partial agreement with users being "neutral" with some of the subcategories predicted by LLM. This result is shown in Table 6.16.
    - We see that users for certain disciplines completely agree with the predicted category and partially with the subcategories. We looked at the labels added by users and observed some of them were very specific, whereas others were very general. Additionally, we observe that the skill level and subject matter expertise of users vary, which can be considered to be a confounding variable in this user study.

2. Our approach of obtaining context from the document abstract and references that correlate with each chapter helps us select a better set of important keywords and sentences and thus create summaries that scored higher by LLM-based evaluation and are preferred by users.

- We see in results reported in Section 6.4.1, using G-Eval, that summary 4, i.e., Llama 2 generated summary aided with context from abstract sentences and reference text, performed better in most metrics.
- Our user study results in Figures 6.6, 6.8, 6.11, 6.12, 6.13, and 6.14 show that users prefer summaries generated by using context from document abstract sentences and reference titles, 66.67% of the times.

3. Our approach to using an ensemble summarization technique that incorporates LLMs generates better summaries that are preferred by users.
    - Our user study results in Figures 6.1, 6.2, and 6.3 show that users consistently prefer summaries generated by using an ensemble method using LLMs.

Chapter 7

Conclusion and Future Work

7.1 Summary

We describe methods to help extract and generate components from book-length scholarly documents. We provide chapter labels and summaries to help readers find the interesting portions faster. This aims to make ETDs more accessible by providing users with more extensive derived metadata. We propose methods to detect interdisciplinarity in ETD chapters by predicting multi-labels. We use a probabilistic method by utilizing the activation function and use LLMs to predict categories and sub-categories. We also propose a summarization pipeline that leverages context from ETD summaries and references to help select important parts from a chapter text. We use a human evaluation framework to evaluate our methods. We observe that users preferred summaries generated by our methods of including context from document abstracts and citing reference titles.

We proposed the following research questions at the beginning of the research work.

1. Can tools that we devise achieve high accuracy in segmenting long documents automatically? This work has been discussed in Chapter 3.

2. Can chapter-level classification labels and chapter summaries help in understanding the chapter text better? This work has been discussed in Chapters 4 and 5.

3. Can leveraging LLMs and obtaining context from document abstract sentences and cited reference titles from each chapter help in creating better chapter summaries? The work has been discussed in Chapters 5 and 6.

4. Can we use human evaluation to create more ground truth data that helps create better models in the future? This work has been discussed in Chapter 6.

We had proposed the following hypotheses.

1. Our approach to automatic chapter boundary detection using a neural network model trained on long scholarly documents yields better results than available SOTA neural network models trained on collections of shorter documents.

    - We find that the neural model trained on scholarly documents yields better results than SOTA models trained on shorter documents, as shown in Section 3.3.2.
    - The neural model generates false positives for chapter boundaries. Therefore, the quality is unacceptable. Hence, we rely on manual segmentation to create the ETD-SGT* datasets as described in Section 3.4.

2. Our quantitative experimental results show that a language model built on clean text (i.e., only complete sentences) will result in a model that better understands the scholarly language, as measured by the perplexity score.

    - We have proved that a language model built on clean text results in models that better understand the scholarly text as discussed in Chapter 4, especially Table 4.7.

3. Our quantitative experimental results show that sophisticated transformer-based language model classifiers outperform traditional machine learning-based classifiers such as Support Vector Machine and Random Forest.

- We have proved that language model-based classification methods outperform traditional machine learning-based classifiers, as proved by the ROC analysis (Appendix Figures B.1, B.1, B.3, B.5, B.7, B.9, and B.11) and F1, Precision, and Recall scores in Chapter 4, Tables 4.4 and 4.5.

4. Our quantitative experimental comparison shows that the classification of our ETD chapters using language models improves when fine-tuning SOTA pre-trained models on our corpus as opposed to using SOTA pre-trained models.

    - We have proved that fine-tuning language models improves the performance of the classification model as discussed in Chapter 4, especially Table 4.6.

5. Our qualitative experimental results show that a multi-label classifier outperforms a multi-class classifier at predicting the correct discipline as measured by F-1, Precision, and Recall scores.

    - Our experimental results show that multi-label classification improved the performance of our fine-tuned classifiers as shown in Chapter 4, especially Table 4.13.

6. Our experiments regarding prompting and fine-tuning LLMs for classification will generate a better understanding of the chapter content, as shown by our user study.

    - Our experimental results show that prompting and instruction tuning Llama 2 and Llama 3 results in the generation of granular classification results in the form of categories and subcategories. Methods of generating results for this hypothesis have been discussed in Sections 4.4.4 and 4.4.5.2. We evaluated the results in Chapter 6 and report the results in Sections 6.4.2 and 6.4.3.

    - Users in the CS study agreed with the LLM-predicted chapter category and subcategories as shown in Table 6.10.

- Users in the non-CS departments agreed with 3 out of 4 categories predicted by the LLM. For subcategories, we observed partial agreement with users being "neutral" with some of the subcategories predicted by the LLM. This result is shown in Table 6.16.
- We see that users for certain disciplines completely agree with the predicted category and partially with the subcategories. We looked at the labels added by users and observed some of them were very specific, whereas others were very general. Additionally, we observe that the skill level and subject matter expertise of users vary, which can be considered to be a confounding variable in our user study.

7. Our experimental results prove that abstractive models with longer context windows generate better summaries than models with shorter context windows as proved by higher ROUGE scores.
   - Our experimental results show that abstractive models generate summaries with higher ROUGE scores than extractive models as shown in Chapter 5, especially Table 5.6.
8. Our approach of obtaining context from the document abstract and references that correlate with each chapter helps us select a better set of important keywords and sentences and thus create summaries that are preferred by users.
   - We see in results reported in Section 6.4.1, that summary 4, i.e., the Llama generated summary aided with context from abstract sentences and reference text, performed better in most metrics.
   - Our user study results in Chapter 6, reported in Figures 6.6, 6.8, 6.11, 6.12, 6.13, and 6.14, show that users prefer summaries generated by using context from

document abstract sentence and reference titles, 66.67% of the time. The other 33.34% of users rated summary 3 higher.

9. Our approach to using an ensemble summarization technique that incorporates LLMs generates better summaries that are preferred by users.

    - Our user study results in Chapter 6, reported in Figures 6.1, 6.2 and 6.3, show that users consistently prefer summaries generated by using an ensemble method using LLMs.

7.2 Addressing Bias and Variability

Researchers need to be aware of bias in datasets and algorithms and the potential impact on society. We tried to collect documents from diverse universities across the United States. We are committed to inclusively and tried to make our dataset representative of not only a variety of disciplines but also to include documents from HBCUs and HSIs in the U.S. as discussed in Chapter 3, especially Section 3.2.1.

The latest advances in AI, although remarkable, come with several challenges. The most important of these challenges is the risk of the model hallucinating and generating factually incorrect responses. Apart from instruction-tuning the model, it is also important to make sure the model is "grounded". Retrieval augmented generation (RAG) [66] has become a standard practice to help models generate relevant responses that are grounded in context from the retrieval. In the RAG process, the first step is retrieval and the second step is generation. This approach is beyond the scope of the current work and therefore left for future research. In the future, a RAG pipeline could aid the model in better understanding the context, and providing helpful explanations.

7.3 Future Work

In the future, we want to continue our user study to analyze generated summary quality across a more diverse set of disciplines. Our current IRB proposal (see Appendix D in Section D.2) is valid until 2026. We plan to continue recruiting participants by posting on various listservs. Additionally, we also plan to conduct several studies in courses spanning different disciplines. For this, we will reach out to VT faculty teaching courses during semesters and ask for their help in evaluating chapter summaries and classification labels.

Based on the results, we want to create a feedback loop where the models learn from human annotation and judgements. This reinforcement learning with a human feedback approach [30] will help models learn better from human-annotated data during the training phase.

Moving ahead I want to incorporate a RAG pipeline with my work on LLMs. I want to develop a RAG-LLM pipeline that helps enhance user interaction with ETDs. My goal is to create a RAG-LLM model by starting with a small-scale prototype with a particular discipline or domain. A locally deployed LLM, fine-tuned on discipline-specific data and grounded by a retrieval architecture, can help readers discover, interact, and understand research better. This pipeline can be scaled as per the needs and demands of additional disciplines.

I am excited to work with researchers across domains and see how this approach can help solve scholarly problems, powered by the advances of artificial intelligence. There has been an immense interest in AI in the past year. However, at the same time, it is important to make sure the models we create are responsible, ethical, and as accurate as possible. I am passionate about making an effort to make AI explainable and something beyond just a 'black box.'

Bibliography

[1] Fox Edward A., William A. Ingram, Aman Ahuja, Satvik Chekuri, and Bipasha Banerjee. 2024. *Structured Documentation Access for Electronic Documents*. U.S. Patent Application 18/585,685; filed February 23, 2024, Virginia.

[2] ABBYY. 2020. ABBYY Cloud OCR SDK. Retrieved January, 2022 from https://www.ocrsdk.com/

[3] ACM. 1960. Computing Classification System. Retrieved January, 2022 from https://dl.acm.org/ccs

[4] Virginia Tech Advanced Research Computing. 2024. Advanced Research Computing. Retrieved May, 2024 from https://arc.vt.edu/content/arc_vt_edu/en/index.html

[5] Aman Ahuja, Alan Devera, and Edward Alan Fox. 2022. Parsing Electronic Theses and Dissertations Using Object Detection. In *Proceedings of the first Workshop on Information Extraction from Scientific Publications*. Association for Computational Linguistics, Online, 121–130. https://aclanthology.org/2022.wiesp-1.14

[6] AI@Meta. 2023. meta-llama/Llama-2-13b-hf · Hugging Face. Retrieved July, 2023 from https://huggingface.co/meta-llama/Llama-2-13b-hf

[7] AI@Meta. 2024. Llama 3 Model Card. Retrieved May, 2024 from https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md

[8] AI@Meta. 2024. meta-llama/Meta-Llama-3-8B-Instruct · Hugging

Face. Retrieved May, 2024 from https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct

[9] Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D., Juan B., and Krys Kochut. 2017. Text Summarization Techniques: A Brief Survey. *International Journal of Advanced Computer Science and Applications* 8, 10 (2017). https://doi.org/10.14569/ijacsa.2017.081052 arXiv:1707.02268

[10] Tor Andersson. 2005. MuPDF. Retrieved February, 2022 from https://mupdf.com/

[11] Anthropic. 2023. Claude 2. Retrieved September, 2023 from https://www.anthropic.com/index/claude-2

[12] John Aromando, Bipasha Banerjee, Bill Ingram, Palakh Mignonne Jude, and Sampanna Kahu. 2019. Classification and extraction of information from ETD documents. http://hdl.handle.net/10919/96645 CS6604, Digital Libraries, Final Report, Virginia Tech, Blacksburg, VA.

[13] Bipasha Banerjee. 2022. Opening scholarly documents through text analytics. In *Proceedings of the 22nd ACM/IEEE Joint Conference on Digital Libraries*. ACM, Online, 1–2. https://doi.org/10.1145/3529372.3530948

[14] Bipasha Banerjee. 2023. IR experimental pilot study option 1: Summarization evaluation, CS5604, Fall 2023, VT. Retrieved May, 2024 from https://canvas.vt.edu/courses/176258/assignments/1927048

[15] Bipasha Banerjee. 2024. Summarization Evaluation, CS4624, Spring 2024, VT. Retrieved May, 2024 from https://canvas.vt.edu/courses/185510/assignments/2034897

[16] Bipasha Banerjee, William A Ingram, Jian Wu, and Edward A Fox. 2022. Applications of data analysis on scholarly long documents. In *2022 IEEE International Conference on Big Data (Big Data)*. IEEE, Online, 2473–2481. doi:10.1109/BigData55660.2022.10020935

[17] Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A Pretrained Language Model for Scientific Text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, 3615–3620. https://doi.org/10.18653/v1/D19-1371

[18] Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The Long-Document Transformer. *arXiv:2004.05150 [cs]* (Dec. 2020). http://arxiv.org/abs/2004.05150 arXiv: 2004.05150.

[19] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. 1992. A Training Algorithm for Optimal Margin Classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory* (Pittsburgh, Pennsylvania, USA) *(COLT '92)*. Association for Computing Machinery, New York, NY, USA, 144–152. https://doi.org/10.1145/130385.130401

[20] Andrew P. Bradley. 1997. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition* 30, 7 (July 1997), 1145–1159. https://doi.org/10.1016/S0031-3203(96)00142-2

[21] Leo Breiman. 2001. Random Forests. *Machine Learning* 45, 1 (01 Oct 2001), 5–32. https://doi.org/10.1023/A:1010933404324

[22] Jason Brownlee. 2020. 4 Types of Classification Tasks in Machine Learn-

ing. Retrieved June 2024 from https://machinelearningmastery.com/types-of-classification-in-machine-learning/

[23] Chiara Campagnola. 2020. Perplexity in Language Models. *Medium* (Oct. 2020). https://towardsdatascience.com/perplexity-in-language-models-87a196019a94

[24] Shuyang Cao and Lu Wang. 2022. HIBRIDS: Attention with Hierarchical Biases for Structure-aware Long Document Summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (Eds.). Association for Computational Linguistics, Dublin, Ireland, 786–807. https://doi.org/10.18653/v1/2022.acl-long.58

[25] Joel Chan, Joseph Chee Chang, Tom Hope, Dafna Shahaf, and Aniket Kittur. 2018. SOLVENT: A Mixed Initiative System for Finding Analogies between Research Papers. *Proc. ACM Hum.-Comput. Interact.* 2, CSCW, Article 31 (Nov 2018), 21 pages. https://doi.org/10.1145/3274300

[26] Harrison Chase. 2022. LangChain. https://github.com/langchain-ai/langchain original-date: 2022-10-17T02:58:36Z.

[27] Satvik Chekuri, Prashant Chandrasekar, Bipasha Banerjee, Sung Hee Park, Nila Masrourisaadat, Aman Ahuja, William A Ingram, and Edward A Fox. 2023. Integrated digital library system for long documents and their elements. In *2023 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*. IEEE, 13–24. https://doi.org/10.1109/JCDL57899.2023.00012

[28] Yinlin Chen. 2017. *A High-quality Digital Library Supporting Computing Education:*

The Ensemble Approach. Ph. D. Dissertation. Virginia Tech. http://hdl.handle.net/10919/78750

[29] Yen Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)* 1, 2017 (2018), 675–686. https://doi.org/10.18653/v1/p18-1063 arXiv:1805.11080

[30] Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2023. Deep reinforcement learning from human preferences. https://doi.org/10.48550/arXiv.1706.03741 arXiv:1706.03741 [cs, stat].

[31] cia. 2023. 2023 List of Minority Serving Institutions (MSIs). https://www.cia.gov/careers/static/8f344010d2bbbcb9bac09c8c9fe40ffc/2023-CMSI-MSI-List-2.pdf. [Accessed 09-05-2024].

[32] Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A Discourse-Aware Attention Model for Abstractive Summarization of Long Documents. http://arxiv.org/abs/1804.05685 arXiv:1804.05685 [cs].

[33] Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel Weld. 2020. SPECTER: Document-level Representation Learning using Citation-informed Transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 2270–2282. https://doi.org/10.18653/v1/2020.acl-main.207

[34] Nancy R. Cook. 2007. Use and Misuse of the Receiver Operating Characteristic Curve in Risk Prediction. *Circulation* 115, 7 (Feb 2007), 928–935. https://doi.org/10.1161/CIRCULATIONAHA.106.672402 Publisher: American Heart Association.

[35] Gianluca Demartini, Jie Yang, and Shazia Sadiq. 2022. Report on the 1st Workshop on Human-in-the-Loop Data Curation (HIL-DC 2022) at CIKM 2022. *ACM SIGIR Forum* 56, 2 (Dec. 2022), 1–8. https://doi.org/10.1145/3582900.3582921

[36] Tim Dettmers. 2024. TimDettmers/bitsandbytes. Retrieved May, 2024 from https://github.com/TimDettmers/bitsandbytes original-date: 2021-06-04T00:10:34Z.

[37] Alan Devera, Raj Sahu, Nila Masrourisaadat, Nirmal Amirthalingam, and Chenyu Mao. 2023. Team 3: Object Detection and Topic Modeling (Objects&Topics) CS 5604 F2022. (Jan. 2023). https://hdl.handle.net/10919/114081 Accepted: 2023-03-10T18:02:15Z Publisher: Virginia Tech.

[38] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[39] Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri. 2022. Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark. http://arxiv.org/abs/2109.14545 arXiv:2109.14545 [cs].

[40] Susan T. Dumais. 2004. Latent semantic analysis. *Annu. Rev. Inf. Sci. Technol.* 38, 1 (2004), 188–230. https://doi.org/10.1002/aris.1440380105

[41] Günes Erkan and Dragomir R. Radev. 2004. LexRank: Graph-based Lexical Centrality as Salience in Text Summarization. *J. Artif. Intell. Res.* 22 (2004), 457–479. https://doi.org/10.1613/jair.1523

[42] Euske. 2010. Welcome to pdfminer.six's documentation! — pdfminer.six 20201018 documentation. Retrieved April, 2023 from https://pdfminersix.readthedocs.io/en/latest/

[43] Kaushik Ganesan, Deepak Nanjundan, Deval Srivastava, Abhilash Neog, Dharneeshkar Jayaprakash, and Aditya Shah. 2023. Team 4: Segmentation, Summarization, and Classification. (Jan. 2023). https://hdl.handle.net/10919/114077 Accepted: 2023-03-10T18:01:06Z Publisher: Virginia Tech.

[44] Google Gemini Team. 2024. Gemini: A Family of Highly Capable Multimodal Models. Retrieved July 2024 from http://arxiv.org/abs/2312.11805 arXiv:2312.11805 [cs].

[45] C. Lee Giles. 2013. Scholarly Big Data: Information Extraction and Data Mining. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management* (San Francisco, California, USA) *(CIKM '13)*. Association for Computing Machinery, New York, NY, USA, 1–2. https://doi.org/10.1145/2505515.2527109

[46] Paul Ginsparg. 1991. arXiv.org e-Print archive. Retrieved March 2023 from https://arxiv.org/

[47] Harshil Goel, Varun Choudhary, Anish Dhondi, and Parth Desai. 2023. *Summarization Evaluation*. Technical Report. Virginia Tech. Retrieved December 2023 from http://hdl.handle.net/10919/115645

[48] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Networks. https://doi.org/10.48550/arXiv.1406.2661 arXiv:1406.2661 [cs, stat].

[49] Alex Graves. 2012. Sequence Transduction with Recurrent Neural Networks. arXiv:1211.3711 Retrieved June 2024 from http://arxiv.org/abs/1211.3711

[50] Komal Gupta, Ammaar Ahmad, Tirthankar Ghosal, and Asif Ekbal. 2021. ContriSci: A BERT-Based Multitasking Deep Neural Architecture To Identify Contribution Statements From Research Papers. In *Towards Open and Trustworthy Digital Societies: 23rd International Conference on Asia-Pacific Digital Libraries, ICADL 2021, Virtual Event, December 1–3, 2021, Proceedings*. Springer-Verlag, Berlin, Heidelberg, 436–452. https://doi.org/10.1007/978-3-030-91669-5_34

[51] J A Hanley and B J McNeil. 1983. A method of comparing the areas under receiver operating characteristic curves derived from the same cases. *Radiology* 148, 3 (Sept. 1983), 839–843. https://doi.org/10.1148/radiology.148.3.6878708

[52] Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching Machines to Read and Comprehend. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett (Eds.). Online, 1693–1701. https://proceedings.neurips.cc/paper/2015/hash/afdec7005cc9f14302cd0474fd0f3c96-Abstract.html

[53] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (Nov. 1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735 eprint: https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf.

[54] Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. 2021. Efficient Attentions for Long Document Summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Lin-*

guistics: Human Language Technologies, Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (Eds.). Association for Computational Linguistics, Online, 1419–1436. https://doi.org/10.18653/v1/2021.naacl-main.112

[55] William Ingram, Edward A. Fox, and Jian Wu. 2019. Opening Books and the National Corpus of Graduate Research. https://www.imls.gov/grants/awarded/lg-37-19-0078-19 Institute of Museum and Library Services Grant LG-37-19-0078-19.

[56] William A Ingram, Bipasha Banerjee, and Edward A Fox. 2019. Summarizing ETDs with deep learning. *Cadernos BAD* 1 (2019), 46–52. https://brapci.inf.br/index.php/res/download/134688

[57] William A. Ingram, Jian Wu, Sampanna Yashwant Kahu, Javaid Akbar Manzoor, Bipasha Banerjee, Aman Ahuja, Muntabir Hasan Choudhury, Lamia Salsabil, Winston Shields, and Edward A. Fox. 2024. Building datasets to support information extraction and structure parsing from electronic theses and dissertations. *International Journal on Digital Libraries* (May 2024). https://doi.org/10.1007/s00799-024-00395-4

[58] Tanya Jain, Hirva Bhagat, Wen-Yu Lee, Ashrith Reddy Thukkaraju, and Raghav Sethi. 2023. CS5604: Team 1 ETD Collection Management. (Jan. 2023). https://hdl.handle.net/10919/114079 Accepted: 2023-03-10T18:01:52Z Publisher: Virginia Tech.

[59] Palakh Mignonne Jude. 2020. Increasing Accessibility of Electronic Theses and Dissertations (ETDs) Through Chapter-level Classification. http://hdl.handle.net/10919/99294. [VTechWorks; VT MS Thesis; Online; accessed 25-September-2020].

[60] Fatih Karabiber. 2021. Binary Classification. Retrieved June 2024 from https://www.learndatasci.com/glossary/binary-classification/

[61] Akshay Kumar. 2016. Efficient Resource Allocation Schemes for Wireless Networks with with Diverse Quality-of-Service Requirements. (Aug. 2016). Retrieved March 2023 from https://hdl.handle.net/10919/87529 [VTechWorks; VT Ph.D. Dissertation; Online].

[62] Sandeep Kumar, Guneet Singh Kohli, Kartik Shinde, and Asif Ekbal. 2022. Team AINLPML @ MuP in SDP 2021: Scientific Document Summarization by End-to-End Extractive and Abstractive Approach. In *Proceedings of the Third Workshop on Scholarly Document Processing*. Association for Computational Linguistics, Gyeongju, Republic of Korea, 285–290. https://aclanthology.org/2022.sdp-1.36

[63] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics* (Sept. 2019), 1234–1240. https://doi.org/10.1093/bioinformatics/btz682

[64] Mina Lee, Megha Srivastava, Amelia Hardy, John Thickstun, Esin Durmus, Ashwin Paranjape, Ines Gerard-Ursin, Xiang Lisa Li, Faisal Ladhak, Frieda Rong, Rose E. Wang, Minae Kwon, Joon Sung Park, Hancheng Cao, Tony Lee, Rishi Bommasani, Michael Bernstein, and Percy Liang. 2022. Evaluating Human-Language Model Interaction. http://arxiv.org/abs/2212.09746 arXiv:2212.09746 [cs].

[65] Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. http://arxiv.org/abs/2104.08691 arXiv:2104.08691 [cs].

[66] Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Vol. 33. Curran Associates, Inc., 9459–9474. https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html

[67] Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*. Association for Computational Linguistics, Barcelona, Spain, 74–81. https://www.aclweb.org/anthology/W04-1013

[68] Ying Lin, Heng Ji, Fei Huang, and Lingfei Wu. 2020. A Joint Neural Model for Information Extraction with Global Features. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 7999–8009. https://doi.org/10.18653/v1/2020.acl-main.713

[69] Ruikai Liu and Jorj McKie. 2020. PyMuPDF: Python bindings for the PDF rendering library MuPDF. Retrieved May 25, 2020 from https://github.com/pymupdf/PyMuPDF

[70] Yang Liu. 2024. nlpyang/geval. Retrieved June 2024 from https://github.com/nlpyang/geval original-date: 2023-05-23T22:05:58Z.

[71] Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment. http://arxiv.org/abs/2303.16634 arXiv:2303.16634 [cs].

[72] Yang Liu and Mirella Lapata. 2019. Text Summarization with Pretrained Encoders. *arXiv:1908.08345 [cs]* (Sept. 2019). http://arxiv.org/abs/1908.08345 arXiv: 1908.08345.

[73] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv* abs/1907.11692 (2019). https://doi.org/10.48550/arXiv.1907.11692

[74] Edward Loper and Steven Bird. 2001. NLTK: nltk.tokenize package. Retrieved June 2024 from https://www.nltk.org/api/nltk.tokenize.html

[75] Patrice Lopez. 2009. GROBID: Combining Automatic Bibliographic Data Recognition and Term Extraction for Scholarship Publications. https://link.springer.com/chapter/10.1007/978-3-642-04346-8_62. In *Research and Advanced Technology for Digital Libraries*, Maristella Agosti, José Borbinha, Sarantos Kapidakis, Christos Papatheodorou, and Giannis Tsakonas (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 473–474.

[76] Fredrik Lundh. 1999. xml.etree.ElementTree — The ElementTree XML API. Retrieved March 2023 from https://docs.python.org/3/library/xml.etree.elementtree.html

[77] Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2024. huggingface/peft. https://github.com/huggingface/peft original-date: 2022-11-25T03:51:09Z.

[78] Javaid Akbar Manzoor. 2023. *Segmenting Electronic Theses and Dissertations By Chapters*. Thesis. Virginia Tech. https://hdl.handle.net/10919/113246 Accepted: 2023-01-19T09:00:28Z.

[79] Charles Martel and J.C.M. Hanson. 1904. Library of Congress Classification Outline - Classification - Cataloging and Acquisitions (Library of Congress). https://www.loc.gov/catdir/cpso/lcco/

[80] Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing Order into Text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, EMNLP 2004, A meeting of SIGDAT, a Special Interest Group of the ACL, held in conjunction with ACL 2004, 25-26 July 2004, Barcelona, Spain*. ACL, 404–411. https://aclanthology.org/W04-3252/

[81] Andreas Mueller. 2016. wordcloud: A little word cloud generator. Retrieved July 2024 from https://github.com/amueller/word_cloud

[82] Kanad Naleshwarkar, Gayatri Bhatambarekar, Zeel Desai, Aishwarya Kumaran, Shadab Haque, and Adithya Harish Srinivasan Manikandan. 2023. Team 4: Language Models, Classification and Summarization. https://hdl.handle.net/10919/118663 Publisher: Virginia Tech.

[83] Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents. *31st AAAI Conference on Artificial Intelligence, AAAI 2017* 1 (2017), 3075–3081. https://doi.org/10.1609/aaai.v31i1.10958

[84] Col Needham. 1990. IMDb Community Forums. Retrieved June 2024 from https://community-imdb.sprinklr.com/

[85] Travis Oliphant. 2006. numpy.argsort — NumPy v1.26 Manual. Retrieved June 2024 from https://numpy.org/doc/stable/reference/generated/numpy.argsort.html

[86] OpenAI. 2020. OpenAI API. Retrieved June 2024 from https://openai.com/index/openai-api/

[87] Manoj Prabhakar Paidiparthy, Ramaraja Ramanujan, Akshita Teegalapally, Madhuvanti Muralikrishnan, Romil Khimraj Balar, Shaunak Juvekar, and Vivek Murali. 2023. *Team 2 for End Users*. Technical Report. Virginia Tech. Retrieved April 2024 from https://hdl.handle.net/10919/114080

[88] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics* (Philadelphia, Pennsylvania) *(ACL '02)*. Association for Computational Linguistics, USA, 311–318. https://doi.org/10.3115/1073083.1073135

[89] Sung Hee Park, Bipasha Banerjee, William A Ingram, and Edward A Fox. 2023. Case Study of Analyzing the Variety of ETD Layouts. In *Proceedings of 26th International Symposium ETD 2023*. INFLIBNET Centre, Gandhinagar. https://ir.inflibnet.ac.in/handle/1944/2414

[90] Priyanka. 2022. Perplexity of Language Models. Retrieved June, 2024 from https://medium.com/@priyankads/perplexity-of-language-models-41160427ed72

[91] ProQuest. 2002. Subject Categories 2019-2020 Academic Year. Retrieved May 2023 from https://about.proquest.com/globalassets/proquest/files/pdf-files/subject-categories-academic.pdf

[92] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018). https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf

[93] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* 21 (2020), 140:1–140:67. http://jmlr.org/papers/v21/20-074.html

[94] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. https://arxiv.org/abs/1908.10084

[95] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B. Tenenbaum, Hugo Larochelle, and Richard S. Zemel. 2018. Meta-Learning for Semi-Supervised Few-Shot Classification. https://doi.org/10.48550/arXiv.1803.00676 arXiv:1803.00676 [cs, stat].

[96] David E. Rumelhart and James L. McClelland. 1987. *Learning Internal Representations by Error Propagation*. MIT Press, 318–362. https://stanford.edu/~jlmcc/papers/PDP/Volume%201/Chap8_PDP86.pdf

[97] SAIG@VT. 2024. Statistical Applications and Innovations Group at Virginia Tech. Retrieved May, 2024 from https://saig.stat.vt.edu/content/saig_stat_vt_edu/en/index.html

[98] A. L. Samuel. 1959. Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development* 3, 3 (1959), 210–229. https://doi.org/10.1147/rd.33.0210

[99] Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM Comput. Surv.* 34, 1 (mar 2002), 1–47. https://doi.org/10.1145/505282.505283

[100] Abi See. 2020. GitHub Repository:abisee/pointer-generator. Retrieved Januay, 2020 from https://github.com/abisee/pointer-generator original-date: 2017-04-23T01:24:56Z.

[101] Amazon Web Services. 2019. OCR Software, Data Extraction Tool - Amazon Textract - AWS. https://aws.amazon.com/textract/

[102] Vedant Shah, Vaishali Ramesh, Reema Daniel, and Mihir D. Gathani. 2023. *Classifying ETDs*. Technical Report. Virginia Tech. Retrieved December 2023 from http://hdl.handle.net/10919/115647

[103] Anmol Shukla, Aaron Travasso, Harish Babu Manogaran, Pallavi Kishor Sisodia, and Yuze Li. 2022. CS5604 Fall 2022 - Team 5 INT. (Jan. 2022). https://hdl.handle.net/10919/114078 Accepted: 2023-03-10T18:01:31Z Publisher: Virginia Tech.

[104] Jeremy Singer-Vine. 2022. pdfplumber. https://github.com/jsvine/pdfplumber original-date: 2015-08-24.

[105] Jake Snell, Kevin Swersky, and Richard S. Zemel. 2017. Prototypical Networks for Few-shot Learning. https://doi.org/10.48550/arXiv.1703.05175 arXiv:1703.05175 [cs, stat].

[106] Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. 2013. Zero-Shot Learning Through Cross-Modal Transfer. In *Advances in Neural Information Processing Systems*, C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger (Eds.), Vol. 26. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2013/file/2d6cc4b2d139a53512fb8cbb3086ae2e-Paper.pdf

[107] Venkat Srinivasan and Edward Fox. 2016. Progress towards automated ETD cataloging. http://docs.ndltd.org/metadata/etd2016/34/index.html 19th Inter-

national Symposium on Electronic Theses and Dissertations (ETD 2016): “Data and Dissertations”.

[108] Ekin Tiu. 2021. Understanding Zero-Shot Learning — Making ML More Human. Retrieved June 2024 from https://towardsdatascience.com/understanding-zero-shot-learning-making-ml-more-human-4653ac35ccab

[109] Dominika Tkaczyk. 2019. What’s your (citations’) style? - Crossref. https://www.crossref.org/blog/whats-your-citations-style/

[110] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *CoRR* abs/2307.09288 (2023). https://doi.org/10.48550/arXiv.2307.09288 arXiv:2307.09288

[111] Sami Uddin, Bipasha Banerjee, Jian Wu, William A. Ingram, and Edward A. Fox. 2021. Building A Large Collection of Multi-domain Electronic Theses and Dissertations. In

2021 IEEE International Conference on Big Data (Big Data), Orlando, FL, USA, December 15-18, 2021, Yixin Chen, Heiko Ludwig, Yicheng Tu, Usama M. Fayyad, Xingquan Zhu, Xiaohua Hu, Suren Byna, Xiong Liu, Jianping Zhang, Shirui Pan, Vagelis Papalexakis, Jianwu Wang, Alfredo Cuzzocrea, and Carlos Ordonez (Eds.). IEEE, Orlando, Florida, USA, 6043–6045. https://doi.org/10.1109/BigData52589.2021.9672058

[112] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[113] Jimmy Wales and Sanger Larry. 2001. Wikipedia. https://en.wikipedia.org/.

[114] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. 2022. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. http://arxiv.org/abs/2207.02696 arXiv:2207.02696 [cs] version: 1.

[115] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned Language Models Are Zero-Shot Learners. http://arxiv.org/abs/2109.01652 arXiv:2109.01652 [cs].

[116] Jeff Wu, Long Ouyang, Daniel M. Ziegler, Nisan Stiennon, Ryan Lowe, Jan Leike, and Paul F. Christiano. 2021. Recursively Summarizing Books with Human Feedback. *CoRR* abs/2109.10862 (2021). arXiv:2109.10862 https://arxiv.org/abs/2109.10862

[117] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang,

and Amr Ahmed. 2020. Big Bird: Transformers for Longer Sequences. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/c8512d142a2d849725f31a9a7a361ab9-Abstract.html

[118] Ruixuan Zhang, Zhuoyu Wei, Yu Shi, and Yining Chen. 2020. BERT-AL: BERT for Arbitrarily Long Document Understanding. (2020). https://doi.org/10.48550/arXiv.1909.11942

[119] Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, and Guoyin Wang. 2024. Instruction Tuning for Large Language Models: A Survey. http://arxiv.org/abs/2308.10792 arXiv:2308.10792 [cs].

[120] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A Survey of Large Language Models. http://arxiv.org/abs/2303.18223 arXiv:2303.18223 [cs].

Appendices

Appendix A

Segmentation Results

In this section, we discuss additional results from Chapter 3, sepecifically related to 'Segmentation'. Tables A.1 and A.2 represent the evaluation of the segmentation models using the arXiv v.s. non-arXiv ETDs using two different embedding models (GloVe and fastText). We look at the performance of the 'Chapter Start' label from non-arXiv documents using two approach variants. Though the F1 score for 'Chapter Start' shown in Table A.1 was higher than that shown in Table A.2, it still was only 46%, making clear there were many errors.

Table A.1: Evaluation for Segmentation Model arXiv vs. non-arXiv (using GloVe embedding). Adapted from [78]

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 A.2: Evaluation for Segmentation Model arXiv vs. non-arXiv (using fastText embedding). Adapted from [78]

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%

Appendix B

Classification Results

In this section, we add more results and additional documents related to Chapter 4.

B.1 ProQuest Mapping

We show the ETD subject to ProQuest subject categories mapping in Figure B.1. The mapping was devised based on the ProQuest subject category system described in [91]. Details of the mapping has been discussed in Chapter 4, Section 4.2.2. We use this mapping to create a classification ground truth dataset, ETD-CL (see Chapter 3, Section 3.5.2), which is used for classification task discussed in Chapter 4. We keep the ETD categories alongside the hierarchical (3 levels) ProQuest departments. ProQuest mapping has a subject code associated with the third level of categories. We also keep the code(s) of the departments.

	ETD_Depts	ProQuest_Depts	Code	Label	Category
2	electrical and computer engineering	electrical engineering,computer engineering	544,464	ENGINEERING	Behavioral, Natural, and Physical Sciences
3	psychology	psychology	621	BEHAVIORAL SCIENCES	Behavioral, Natural, and Physical Sciences
4	mechanical engineering	mechanical engineering	548	ENGINEERING	Behavioral, Natural, and Physical Sciences
5	computer science	computer science	984	ENGINEERING	Behavioral, Natural, and Physical Sciences
6	civil architectural and environmental engineering	civil engineering,architectural engineering,environmental e	543,462,775	ENGINEERING | ARCHITECTURE | ENVIRONMENTAL SCIENCES	Behavioral, Natural, and Physical Sciences | Arts, Business, Education, Humanities, and Social Sciences
7	chemistry	chemistry	485	MATHEMATICAL AND PHYSICAL SCIENCES	Behavioral, Natural, and Physical Sciences
8	physics	physics	605	MATHEMATICAL AND PHYSICAL SCIENCES	Behavioral, Natural, and Physical Sciences
9	mathematics	mathematics	405	MATHEMATICAL AND PHYSICAL SCIENCES	Behavioral, Natural, and Physical Sciences
10	geological sciences	geology	372	GEOSCIENCES	Behavioral, Natural, and Physical Sciences
11	architecture	architecture	729	ARCHITECTURE	Arts, Business, Education, Humanities, and Social Sciences
12	biology	biology	306	BIOLOGICAL SCIENCES	Behavioral, Natural, and Physical Sciences
13	chemical engineering	chemical engineering	542	ENGINEERING	Behavioral, Natural, and Physical Sciences
14	economics	economics	501	SOCIAL SCIENCES	Arts, Business, Education, Humanities, and Social Sciences
15	petroleum and geosystems engineering	petroleum engineering,geological engineering	765,466	ENGINEERING	Behavioral, Natural, and Physical Sciences
16	materials science and engineering	materials science	794	MATHEMATICAL AND PHYSICAL SCIENCES	Behavioral, Natural, and Physical Sciences
17	aerospace engineering	aerospace engineering	538	ENGINEERING	Behavioral, Natural, and Physical Sciences
18	biomedical engineering	biomedical engineering	541	ENGINEERING	Behavioral, Natural, and Physical Sciences
19	animal science	animal sciences	475	AGRICULTURE	Behavioral, Natural, and Physical Sciences
20	earth and atmosphere sciences	atmospheric sciences	725	GEOSCIENCES	Behavioral, Natural, and Physical Sciences
21	home economics education	home economics education	278	EDUCATION	Arts, Business, Education, Humanities, and Social Sciences
22	crop science	plant sciences	479	AGRICULTURE	Behavioral, Natural, and Physical Sciences
23	political science	political science	615	SOCIAL SCIENCES	Arts, Business, Education, Humanities, and Social Sciences
24	kinesiology and health education	kinesiology,health education	575,680	HEALTH AND MEDICAL SCIENCES | EDUCATION	Behavioral, Natural, and Physical Sciences | Arts, Business, Education, Humanities, and Social Sciences
25	linguistic	linguistics	290	LANGUAGE AND LITERATURE	Arts, Business, Education, Humanities, and Social Sciences
26	communication sciences and disorders	communication,disability studies	459,205	COMMUNICATIONS AND INFORMATION SCIENCES | INTERDISCIPLINARY	Arts, Business, Education, Humanities, and Social Sciences
27	education	education	515	EDUCATION	Arts, Business, Education, Humanities, and Social Sciences
28	music	music	413	FINE AND PERFORMING ARTS	Arts, Business, Education, Humanities, and Social Sciences
29	english	english literature	593	LANGUAGE AND LITERATURE	Arts, Business, Education, Humanities, and Social Sciences
30	curriculum and instruction	curriculum development	727	EDUCATION	Arts, Business, Education, Humanities, and Social Sciences
31	history	history	578	HISTORY	Arts, Business, Education, Humanities, and Social Sciences
32	anthropology	cultural anthropology	326	SOCIAL SCIENCES	Arts, Business, Education, Humanities, and Social Sciences
33	sociology	sociology	626	SOCIAL SCIENCES	Arts, Business, Education, Humanities, and Social Sciences
34	communication studies	communication	459	COMMUNICATIONS AND INFORMATION SCIENCES	Arts, Business, Education, Humanities, and Social Sciences
35	radio-television-film	film studies	900	FINE AND PERFORMING ARTS	Arts, Business, Education, Humanities, and Social Sciences
36	community and regional planning	area planning and development	341	SOCIAL SCIENCES	Arts, Business, Education, Humanities, and Social Sciences
37	business administration	business administration	310	BUSINESS	Arts, Business, Education, Humanities, and Social Sciences
38	government	political science	615	SOCIAL SCIENCES	Arts, Business, Education, Humanities, and Social Sciences
39	journalism	journalism	391	COMMUNICATIONS AND INFORMATION SCIENCES	Arts, Business, Education, Humanities, and Social Sciences
40	advertising	marketing	338	BUSINESS	Arts, Business, Education, Humanities, and Social Sciences
41	geography	geography	366	SOCIAL SCIENCES	Arts, Business, Education, Humanities, and Social Sciences
42	theatre and dance	theater,dance	465,378	FINE AND PERFORMING ARTS	Arts, Business, Education, Humanities, and Social Sciences
43	social work	social work	452	SOCIAL SCIENCES	Arts, Business, Education, Humanities, and Social Sciences
44	philosophy	philosophy	422	PHILOSOPHY AND RELIGION	Arts, Business, Education, Humanities, and Social Sciences
45	counselling leadership adult education and school p	counseling psychology,educational leadership,adult educa	603, 449, 516, 519	BEHAVIORAL SCIENCES | EDUCATION	Behavioral, Natural, and Physical Sciences | Arts, Business, Education, Humanities, and Social Sciences
46	spanish italian and portuguese	language,italian studies	679,222	LANGUAGE AND LITERATURE	Arts, Business, Education, Humanities, and Social Sciences
47	art design and history	fine arts,art history,design	357,377,389	FINE AND PERFORMING ARTS	Arts, Business, Education, Humanities, and Social Sciences
48	public affairs and policy	public administration,public policy	617,630	SOCIAL SCIENCES	Arts, Business, Education, Humanities, and Social Sciences

B.2 System Details

In this section, we report the machine details for running the machine learning and language model-based classifiers. It is important to note that machine-learning-based classifiers take significantly less time than language model-based classifiers. Train and inference time and model capacity are definitely advantages for ML-based classifiers such as SVM and RF. The models are known to be lightweight and thus easy to use without having to require sophisticated resources. However, LM-based classifiers outperform in terms of quality measures, making them a better choice for the task.

Table B.1: System details depicting model capacity and time taken for train and inference on PQDT dataset (see Section 3.5.1)

Model	**Task**	**Machine Details**	**Time**
SVM	Classification training and inference	CPU: Intel i7-9700 CPU 3.00GHz, 8 cores GPU: Nvidia Quadro RTX 4000, 8GB GPU memory	1 hour
RF	Classification training and inference	CPU: Intel i7-9700 CPU 3.00GHz, 8 cores GPU: Nvidia Quadro RTX 4000, 8GB GPU memory	0.21 hours
Autoregressive LMs (e.g., BERT)	Fine-tuning	CPU: AMD EPYC 7452, 32 cores GPU: NVIDIA A10 24 GB	~48 hours
Autoregressive LMs (e.g., BERT)	Classification training and inference	CPU: AMD EPYC 7452, 32 cores GPU: NVIDIA A10, 24 GB	2-8 hours (based on model size)

B.3 ROC Curves

This section discusses ROC curve results for SVM, RF, BERT. SciBERT, BERT+ETD, SciBERT+ETD. We use the receiver operating characteristic curve to have a better look at the performance of classifiers at several thresholds. We use experiments described in Chapter 4, specifically the ones described in Tables 4.4 and 4.5. Figures B.1, B.2, B.3, and B.4 depict the ROC curves of the SVM model and for multiple classes for two different experimental setups. Similarly, Figures B.5, B.6, B.7, and B.8 depict the ROC curves of the Random Forest (RF) model and for multiple classes for two different experimental setups with different hyperparameters. We see that the SVM model performs slightly better than RF, but both have an area under the curve of 0.5. We also plot the ROC curves for language models as depicted in Figures B.9 and B.10, for BERT, fine-tuned on ETDs and in Figures B.3 and B.4, for SciBERT fine-tuned on ETDs. Both language model-based classifiers have ROC curve area at 0.98.

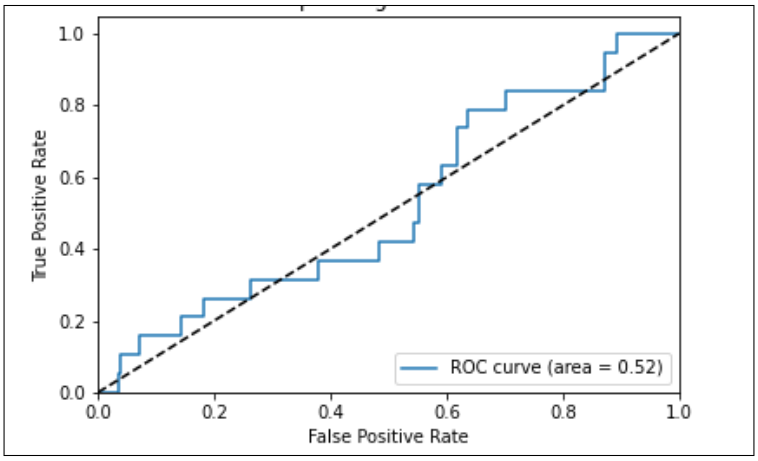
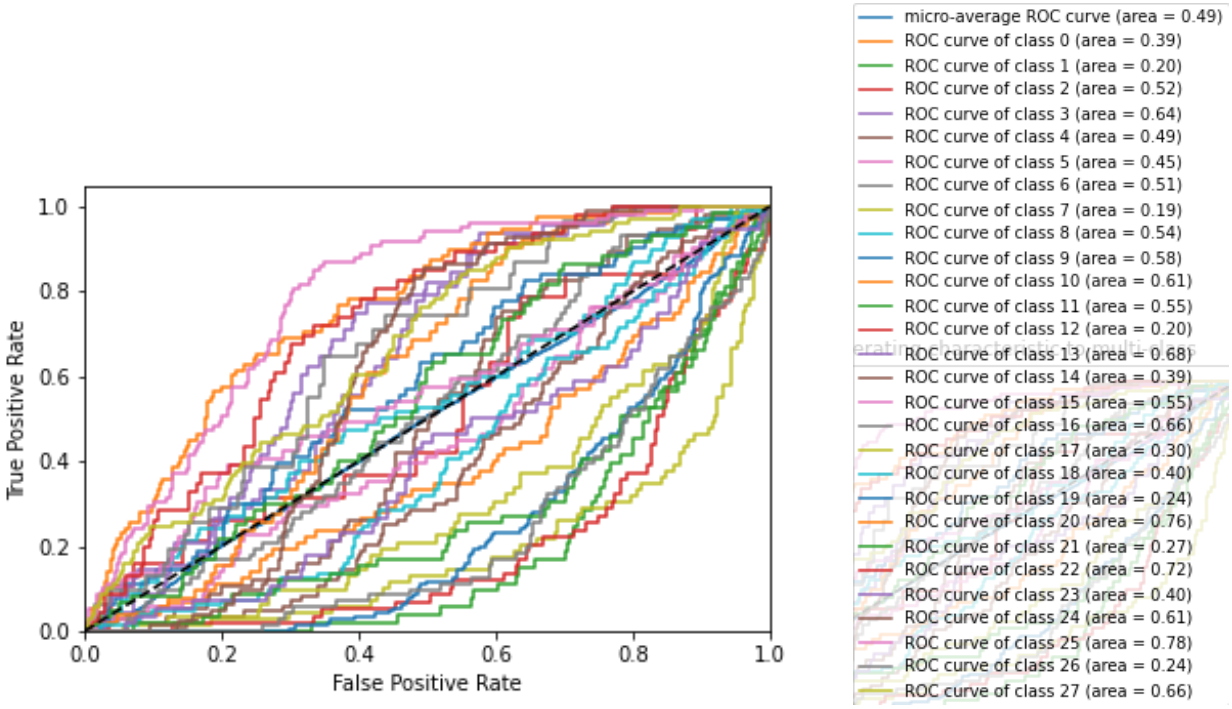


Figure B.1: SVM ROC Curve on PQDT dataset - setup 1



ROC analysis multi-class Legend

Figure B.2: SVM ROC analysis for multiple classes - PQDT dataset - setup 1

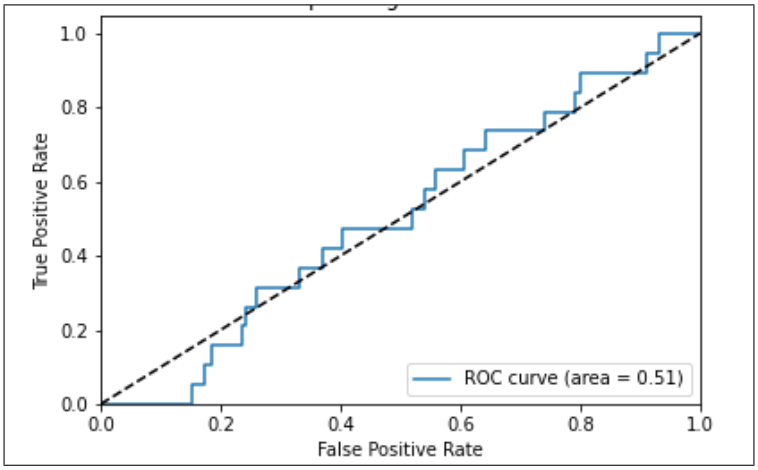
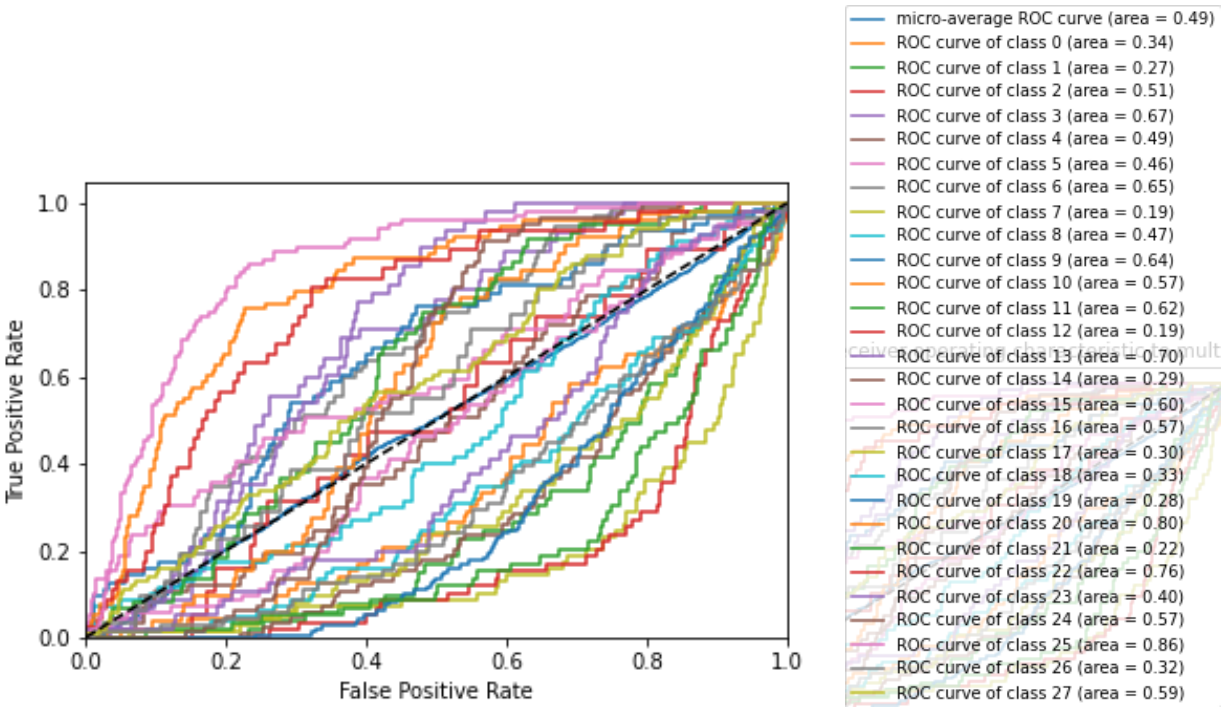


Figure B.3: SVM ROC Curve on PQDT dataset - setup 2



ROC analysis multi-class Legend

Figure B.4: SVM ROC analysis for multiple classes - PQDT dataset - setup 2

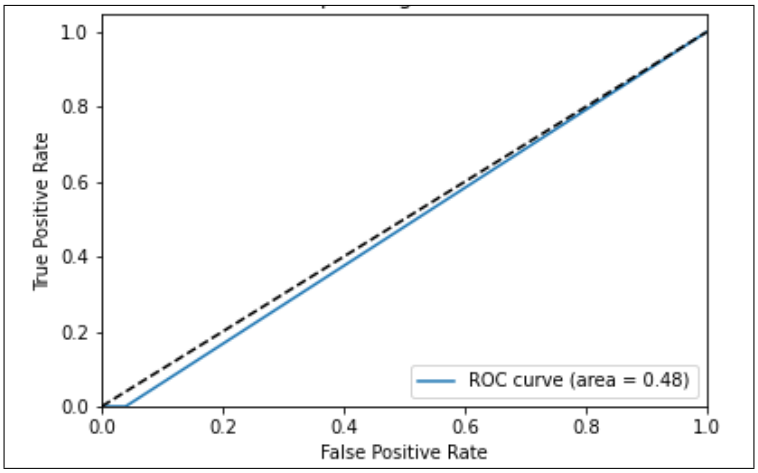
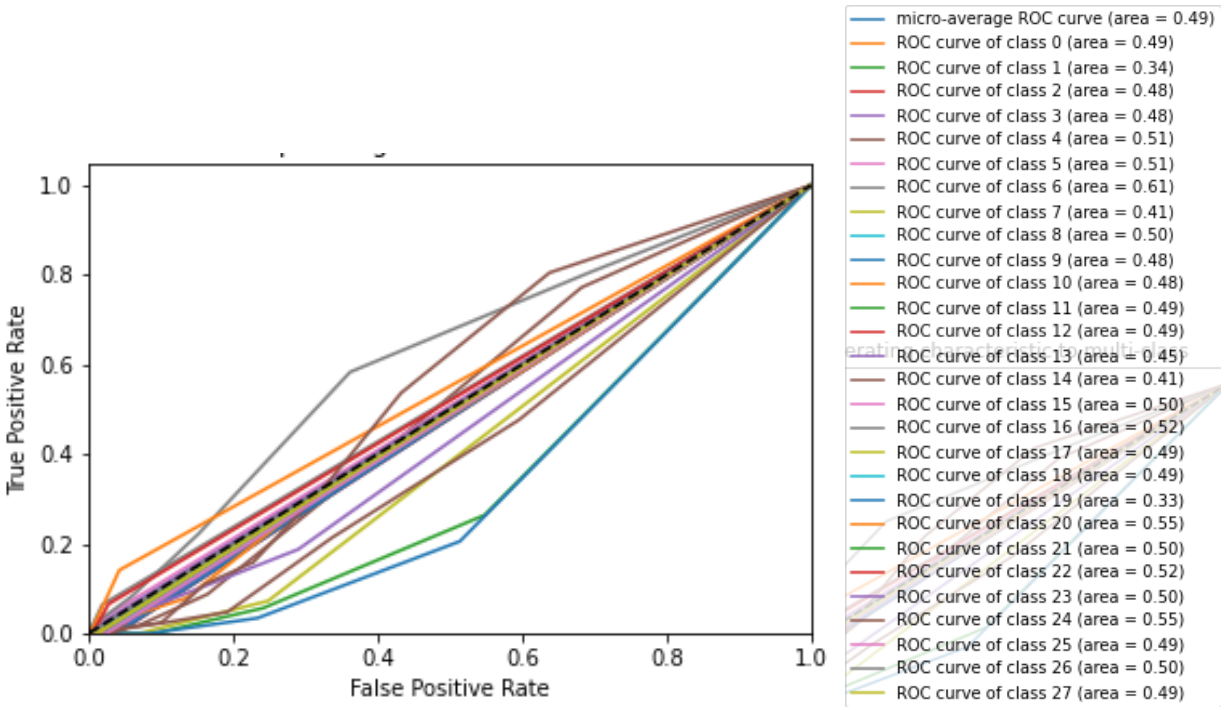


Figure B.5: RF ROC Curve on PQDT dataset - setup 1

B.4 Categories Predicted by Llama

We use Llama 3 for classification of subject categories as discussed in Chapter 4, Section 4.3.3 using the ETD-CL dataset in Section 3.5.2. The prediction is obtained by prompting the



ROC analysis multi-class Legend

Figure B.6: RF ROC analysis for multiple classes - PQDT dataset - setup 1

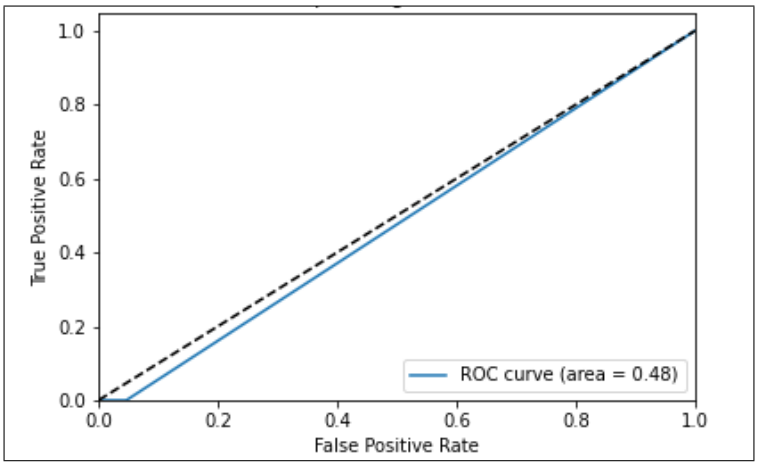
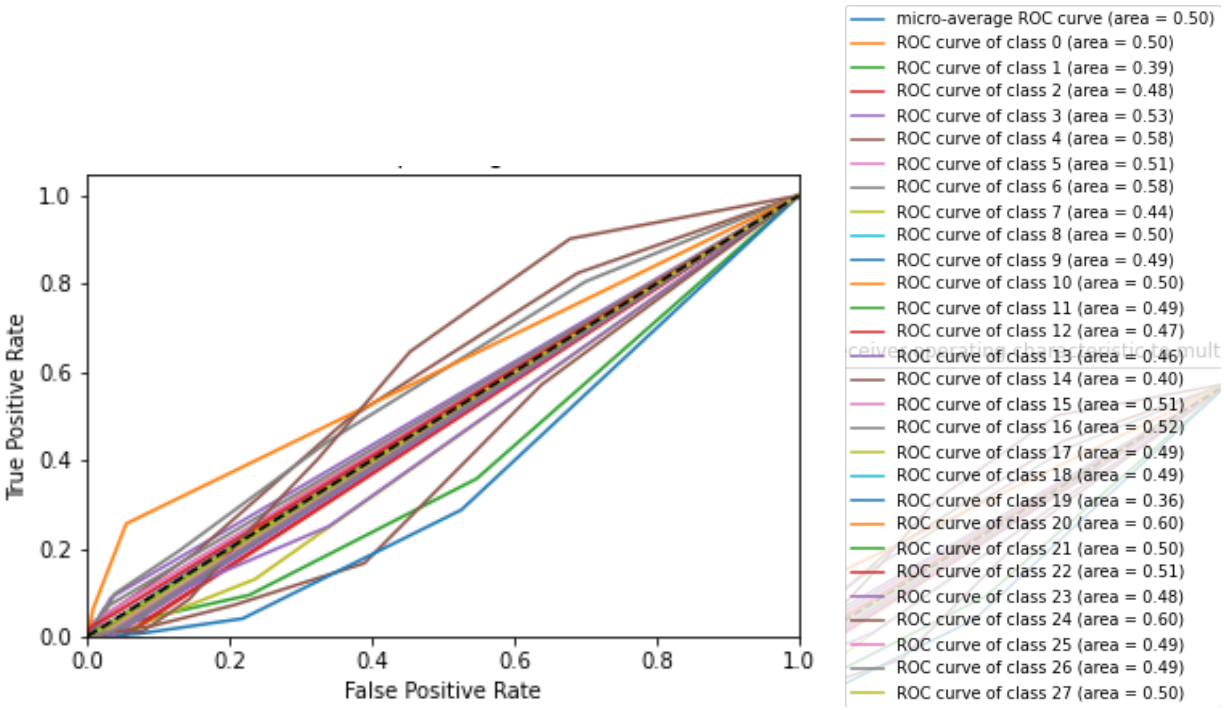


Figure B.7: RF ROC Curve on PQDT dataset - setup 2

Llama 3 model as shown in Figure 4.6. The prompt is designed to output categories and subcategories. Llama 3 predicted a total of 82 categories; however, the input prompt only had 47 categories. We report all the classes (categories) that were predicted by Llama 3 in



ROC analysis multi-class Legend

Figure B.8: RF ROC analysis for multiple classes - PQDT dataset - setup 2

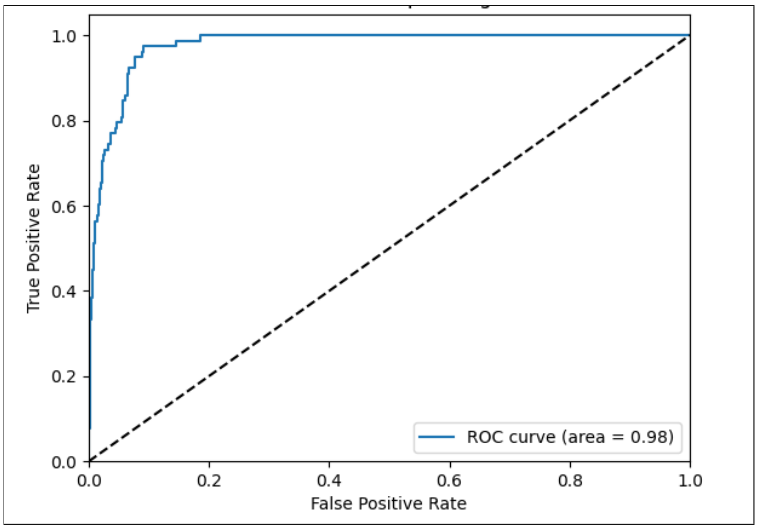
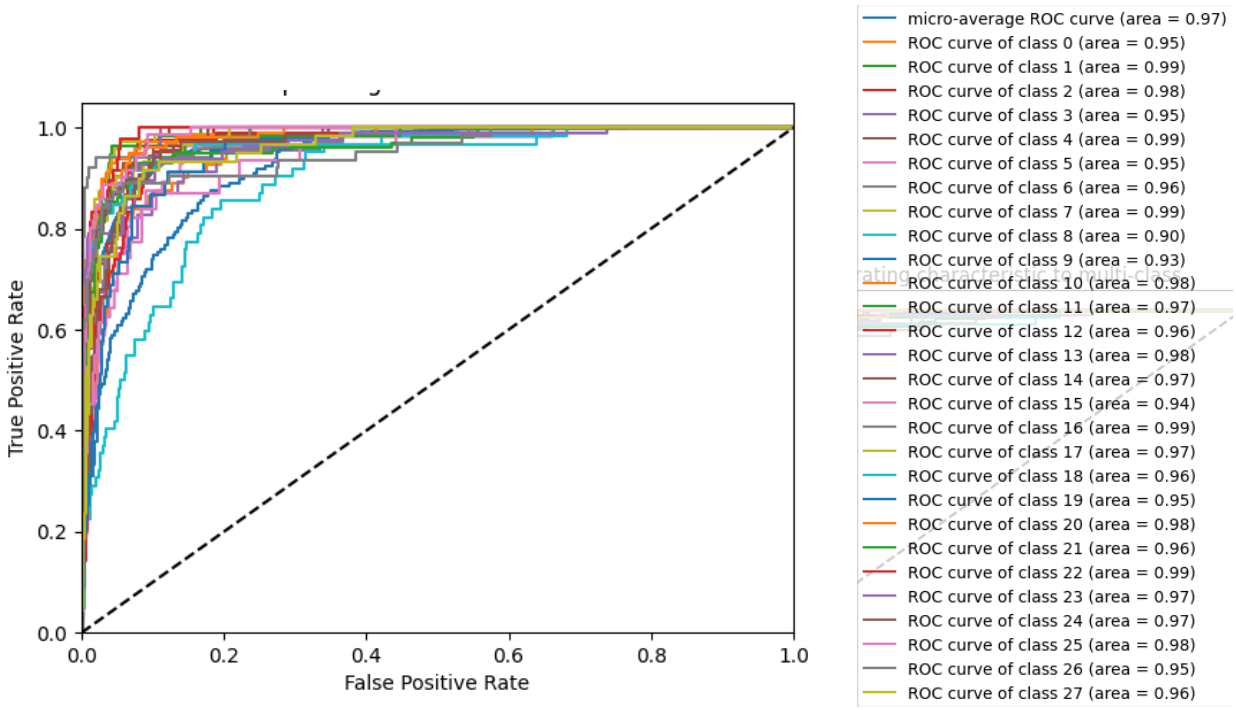


Figure B.9: Finetuned-BERT ROC Curve on PQDT dataset

Table B.2.



ROC analysis multi-class Legend

Figure B.10: Finetuned-BERT ROC analysis for multiple classes - PQDT dataset

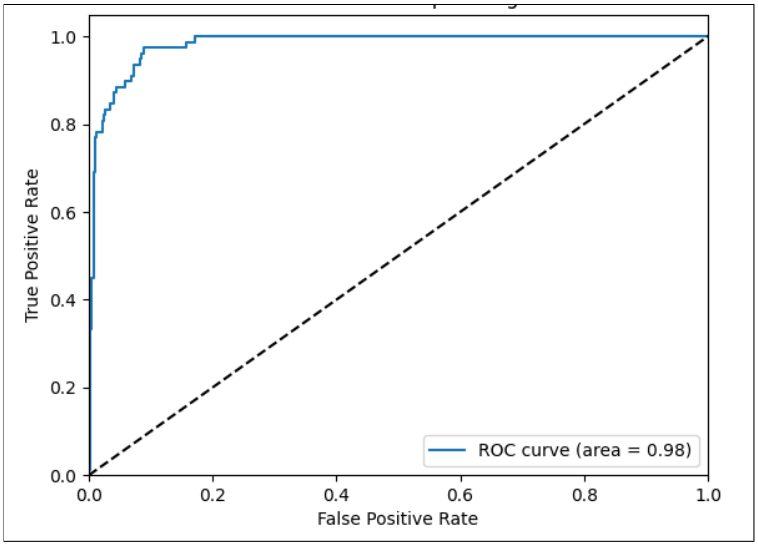


Figure B.11: Finetuned-SciBERT ROC Curve on PQDT dataset

Table B.2: Categories predicted by Llama 3 model

Category Predicted	No. of documents
psychology	174

Category Predicted	No. of documents
computer science	126
geological sciences	114
communication sciences and disorders	110
biomedical engineering	70
petroleum and geosystems engineering	63
electrical and computer engineering	59
civil architectural and environmental engineering	59
linguistic	55
chemical engineering	51
communication studies	43
music	42
materials science and engineering	40
economics	38
chemistry	38
mechanical engineering	37
curriculum and instruction	37
education	36
philosophy	35
counselling leadership adult education and school psychology	34
political science	33
aerospace engineering	32
geography	31
biological sciences	31
biology	28

Category Predicted	No. of documents
mathematics	28
anthropology	26
history	22
architecture	21
animal science	21
business administration	20
theatre and dance	17
art design and history	17
public affairs and policy	16
social work	16
english	15
government	14
sociology	12
literary studies	12
physics	12
kinesiology and health education	11
urban planning	11
international relations	8
literature	7
crop science	7
community and regional planning	6
agricultural sciences	6
public health	5
radio-television-film	5

Category Predicted	No. of documents
archaeology	4
earth and atmosphere sciences	4
law	4
art history	3
accounting	3
literary criticism	3
ecological sciences	2
environmental engineering	2
healthcare administration	1
conservation biology	1
geopolitics	1
dance studies	1
engineering	1
classical studies	1
architectural and environmental engineering	1
criminology	1
nursing	1
agricultural science	1
screenwriting	1
performance studies	1
politics and international relations	1
classics	1
rhetoric and composition	1
law and government	1

Category Predicted	No. of documents
organizational behavior	1
food science	1
home economics education	1
social sciences	1
urban ecology	1
sustainable consumption research and action initiative	1
cosmology	1
industrial and organizational psychology	1

B.5 Top 3 Prediction: Category-wise Results

In this section, we report the performance of top 3 predictions for each of the classes using fine-tuned language models on the ETD-CL dataset. The experiment setup and results have been discussed in Chapter 4, Sections 4.3.4.1 and 4.4.5.1. We report the performance of each class for top 3 predicting models as discussed in Chapter 4, Table 4.13.

Table B.3: Performance of each class using fine-tuned BERT

Category	Accuracy
Advertising	0.85
Aerospace engineering	0.75
Animal science	0.90
Anthropology	0.75
Architecture	0.75

Category	Accuracy
Art and design	0.85
Biology	1.00
Biomedical engineering	0.75
Business administration	0.95
Chemical engineering	0.85
Chemistry	0.70
Civil architectural and environmental engineering	0.75
Communication studies	1.00
Community and regional planning	1.00
Computer science	0.85
Counselling leadership adult	
Education and school psychology	0.60
Curriculum and instruction	0.95
Economics	0.95
Education	0.75
Electrical and computer engineering	0.90
English	0.90
Geography	0.90
Geological sciences	0.95
Government	0.85
History	0.95
Journalism	0.70
Materials science and engineering	0.60
Mathematics	0.85

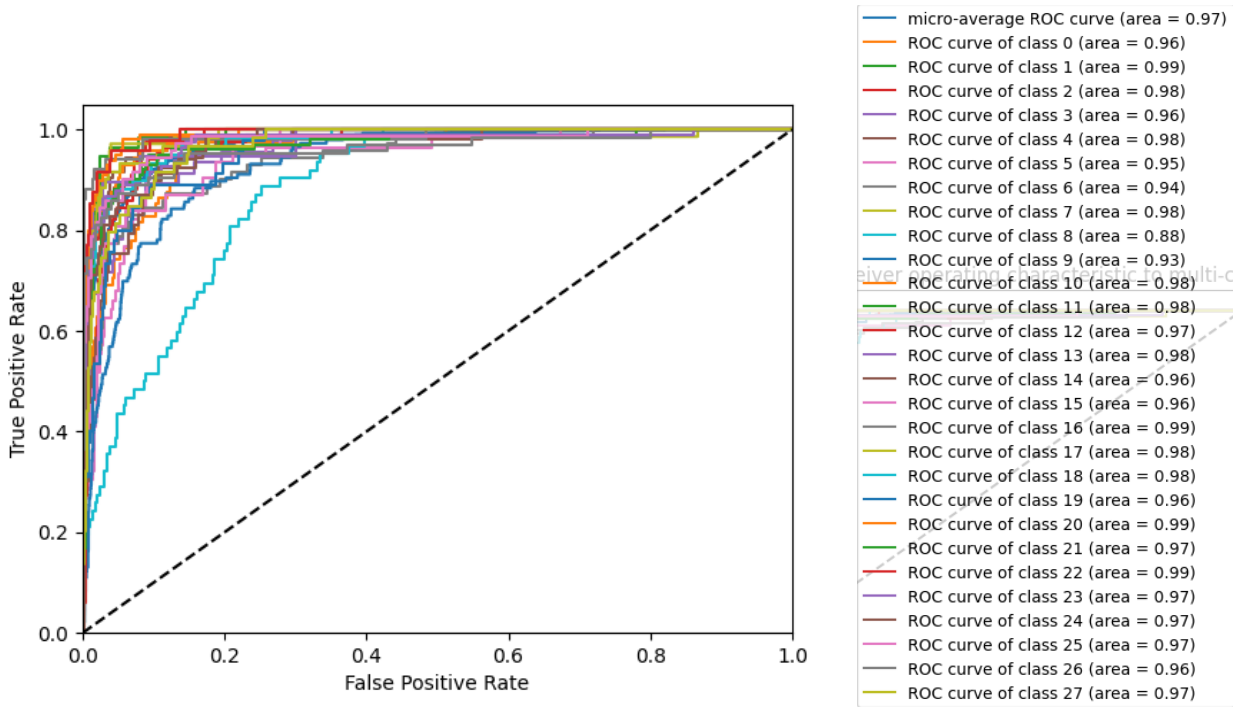
Category	Accuracy
Mechanical engineering	0.65
Music	1.00
Nuclear plasma and radio engineering	0.90
Petroleum and geosystems engineering	0.95
Philosophy	0.95
Physics	0.85
Psychology	0.80
Public affairs	0.80
Radio-television-film	1.00
Social work	0.80
Sociology	0.70
Special education	1.00
Theatre and dance	0.95
accuracy	0.84

Table B.4: Performance of each class using fine-tuned SciBERT

Category	Accuracy
Advertising	0.88
Aerospace engineering	0.88
Animal science	0.98
Anthropology	0.83
Architecture	0.85
Art design and history	0.83

Category	Accuracy
Biology	0.95
Biomedical engineering	0.85
Business administration	0.95
Chemical engineering	0.90
Chemistry	0.98
Civil architectural and environmental engineering	0.93
Communication sciences and disorders	0.95
Communication studies	0.80
Community and regional planning	0.95
Computer science	0.95
Counselling leadership adult education and school psychology	0.90
Crop science	0.85
Curriculum and instruction	0.83
Earth and atmosphere sciences	0.93
Economics	0.98
Education	0.98
Electrical and computer engineering	0.98
English	0.95
Geography	0.83
Geological sciences	0.98
Government	0.91
History	0.93
Home economics education	1.00
Journalism	0.80

Category	Accuracy
Kinesiology and health education	0.98
Linguistic	0.93
Materials science and engineering	0.98
Mathematics	0.95
Mechanical engineering	0.78
Music	1.00
Petroleum and geosystems engineering	1.00
Philosophy	0.83
Physics	0.85
Political science	0.85
Psychology	0.93
Public affairs and policy	0.90
Radio-television-film	0.83
Social work	0.98
Sociology	0.88
Spanish italian and portuguese	0.85
Theatre and dance	0.93
Accuracy	0.91



ROC analysis multi-class Legend

Figure B.12: Finetuned-SciBERT ROC analysis for multiple classes - PQDT dataset

B.6 Keywords Added by Users for Classification CS Pilot Study

In this section, we report the word cloud of keywords that were added by users in the CS pilot study (see Chapter 6, Section 6.3.3.2). We use the complete list of keywords, as added by the users for both of the computer science user studies. We use the Wordcloud [81] Python package for this task. We display the word clouds in Figures B.13 and B.14.

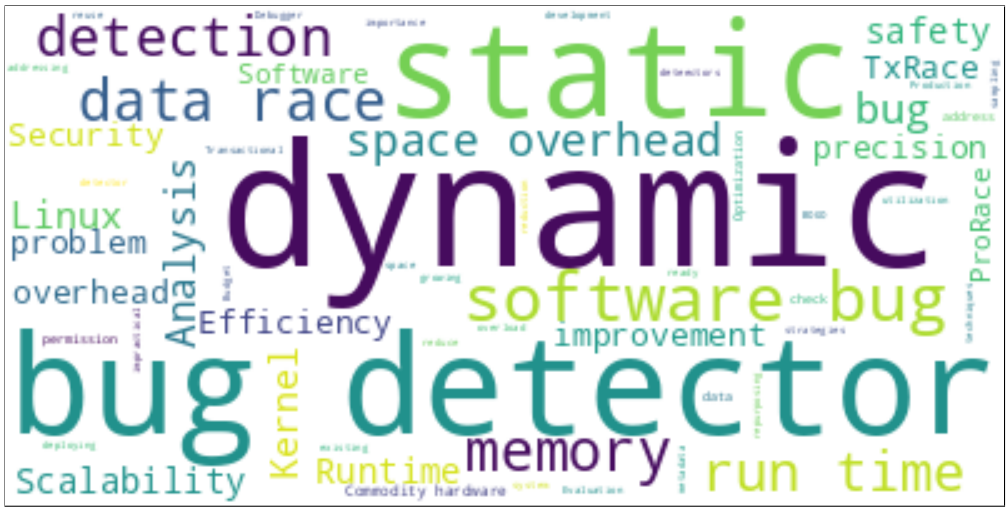


Figure B.13: CS study 1 - wordcloud of keywords

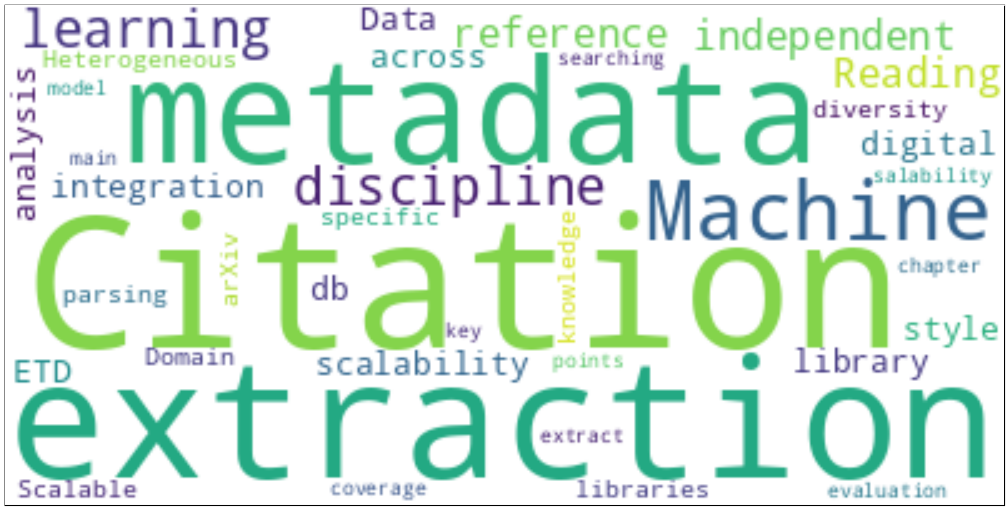


Figure B.14: CS study 2 - wordcloud of keywords

Appendix C

Summarization

C.1 Llama Prompts for Zero-shot and Few-shot Learning

In this section, we show an example of summarization using the Llama 3 model. In the following figures, we discuss the hyperparameters used for the Llama 3 model, zero-shot and few-shot prompts, and results. We first look at the recommended hyperparameters in Figure C.1 as provided by Meta [7]. We change the max_length and device_map information to fit the model in the available GPU. Similarly, for zero-shot, few shot prompts discussed in Figures C.2, C.4, we use the prompts provided by Meta in [7] and modify from there. Some of the prompt modification comes from trial and error approach. This involves modifying the prompts to guide the model to output very specific results such as "output the summaries in bullet points or a paragraph." The results from zero-shot and few-shot prompting Llama 3 models have been discussed in Figures C.3 and C.5.

C.2 Prompts for Using LLM for Summary Evaluation

We show the prompts used for evaluating summaries using GPT-4. The base prompt information has been adapted from the original paper [71]. We modify the prompts to use ETDs

```python
import transformers
import torch

model_id = "meta-llama/Meta-Llama-3-8B-Instruct"

pipeline = transformers.pipeline(
    "text-generation",
    model=model_id,
    model_kwargs={"torch_dtype": torch.bfloat16},
    device_map="auto",
    max_length=8000,
    do_sample=True,
    top_k=10,
)
```

Figure C.1: Llama 3 model hyperparameters for summarization

as a summary evaluation dataset.

```python
messages = [
    {"role": "system", "content": "Write a concise summary of the  text provided. Return your response \
    in a paragraph"},
    {"role": "user", "content": text},

]

prompt = pipeline.tokenizer.apply_chat_template(
        messages,
        tokenize=False,
        add_generation_prompt=True
)

terminators = [
    pipeline.tokenizer.eos_token_id,
    pipeline.tokenizer.convert_tokens_to_ids("<|eot_id|>")
]

outputs = pipeline(
    prompt,
    max_new_tokens=1024,
    eos_token_id=terminators,
    do_sample=True,
    temperature=0.6,
    top_p=0.9,
)
print(outputs[0]["generated_text"][len(prompt):])
```

Figure C.2: Llama 3 summarization zero shot prompt example

```
Here is a concise summary of the text in a paragraph:

This thesis aims to extend the accessibility of knowledge graphs to non-expert users by solving four subproblems: effective logical reasonin
g, retrieval through reasoning, integrating semantic and structural information, and scaling hyperbolic architectures. The research introduce
s several models, including Hyperboloid Embeddings (HypE) for logical reasoning, ANTHEM for natural language queries, TESH-GCN for hybrid mul
timodal reasoning, and H-GRAM for meta-learning. The thesis concludes by highlighting the potential of hyperbolic space in machine learning,
but also identifies several challenges that need to be addressed, including the development of more complex objective functions, stable train
ing techniques, and scalable formulations. Future research directions include exploring the potential of hyperbolic networks in natural langu
age processing, computer vision, and network analysis.
```

Figure C.3: Llama 3 summarization zero shot prompt result

```
#few shot summarization
messages = [
     {"role": "system", "content": "Please summarize the given text into a paragraph. Few examples are provided below"},
     {"role": "system", "content": "Prompt engineering is a technique used in natural language processing (NLP) to improve \
     the performance of the language model by providing them with more context and information about the task in hand. \
     It involves creating prompts, which are short pieces of text that provide additional information or guidance \
     to the model, such as the topic or genre of the text it will generate. By using prompts, the model can better \
     understand what kind of output is expected and produce more accurate and relevant results. In Llama 2 the size \
     of the context, in terms of number of tokens, has doubled from 2048 to 4096."},
    {"role": "system", "content": "Llama 3 is a powerful LLM capable of providing longer context length"},

    {"role": "system", "content": "Summarize the content in the following text in a single paragraph "},
    {"role": "user", "content": text},

]

prompt = pipeline.tokenizer.apply_chat_template(
        messages,
        tokenize=False,
        add_generation_prompt=True
)

terminators = [
    pipeline.tokenizer.eos_token_id,
    pipeline.tokenizer.convert_tokens_to_ids("<|eot_id|>")
]

outputs = pipeline(
    prompt,
    max_new_tokens=1024,
    eos_token_id=terminators,
    do_sample=True,
    temperature=0.6,
    top_p=0.9,
)
print(outputs[0]["generated_text"][len(prompt):])
```

Figure C.4: Llama 3 summarization few shot prompt example

```
Here is a summarized paragraph:

This thesis aims to improve the accessibility of knowledge graphs to non-expert users by solving four subproblems. The main contributions inc
lude the development of Hyperboloid Embeddings (HypE) for logical reasoning, AtteNTive Hyperbolic Entity Model (ANTHEM) for natural language
queries, Text Enriched Sparse Hyperbolic Graph Convolutional Networks (TESH-GCN) for integrating semantic and structural information, and Hyp
erbolic GRAph Meta Learner (H-GRAM) for scaling hyperbolic models over large graphs. The thesis also discusses future work, including the dev
elopment of more complex objective functions, stable training techniques, and scalable formulations, as well as the potential applications of
hyperbolic space in various domains such as natural language processing, computer vision, and network analysis.
```

Figure C.5: Llama 3 summarization few shot prompt result

```
You will be given one summary written for a chapter of an electronic thesis and dissertation.

Your task is to rate the summary on one metric.

Please make sure you read and understand these instructions carefully. Please keep this document open while reviewing, and refer to it as needed.

Evaluation Criteria:

Coherence (1-5) - the collective quality of all sentences. We align this dimension with the DUC quality question of structure and coherence whereby
"the summary should be well-structured and well-organized. The summary should not just be a heap of related information, but should build from
sentence to a coherent body of information about a topic."

Evaluation Steps:

1. Read the chapter text carefully and identify the main topic and key points.
2. Read the summary and compare it to the chapter text. Check if the summary covers the main topic and key points of the chapter text, and if it
presents them in a clear and logical order.
3. Assign a score for coherence on a scale of 1 to 5, where 1 is the lowest and 5 is the highest based on the Evaluation Criteria.


Example:


Source Text:

{{Document}}

Summary:

{{Summary}}


Evaluation Form (scores ONLY):

- Coherence:
```

Figure C.6: Prompt to evaluate the coherence of generated summaries using LLM

```
You will be given one summary written for a chapter of an electronic thesis and dissertation.

Your task is to rate the summary on one metric.

Please make sure you read and understand these instructions carefully. Please keep this document open while reviewing, and refer to it as needed.


Evaluation Criteria:

Consistency (1-5) - the factual alignment between the summary and the summarized source. A factually consistent summary contains only statements
that are entailed by the source document. Annotators were also asked to penalize summaries that contained hallucinated facts.

Evaluation Steps:

1. Read the chapter text carefully and identify the main facts and details it presents.
2. Read the summary and compare it to the chapter text. Check if the summary contains any factual errors that are not supported by the chapter text.
3. Assign a score for consistency based on the Evaluation Criteria.


Example:


Source Text:

{{Document}}

Summary:

{{Summary}}


Evaluation Form (scores ONLY):

- Consistency:
```

Figure C.7: Prompt to evaluate the consistency of generated summaries using LLM

```
You will be given one summary written for a chapter of an electronic thesis and dissertation.

Your task is to rate the summary on one metric.

Please make sure you read and understand these instructions carefully. Please keep this document open while reviewing, and refer to it as needed.


Evaluation Criteria:

Fluency (1-3): the quality of the summary in terms of grammar, spelling, punctuation, word choice, and sentence structure.

- 1: Poor. The summary has many errors that make it hard to understand or sound unnatural.
- 2: Fair. The summary has some errors that affect the clarity or smoothness of the text, but the main points are still comprehensible.
- 3: Good. The summary has few or no errors and is easy to read and follow.


Example:

Summary:

{{Summary}}


Evaluation Form (scores ONLY):

- Fluency (1-3):
```

Figure C.8: Prompt to evaluate the fluency of generated summaries using LLM

```
You will be given one summary written for a chapter of an electronic thesis and dissertation.

Your task is to rate the summary on one metric.

Please make sure you read and understand these instructions carefully. Please keep this document open while reviewing, and refer to it as needed.

Evaluation Criteria:

Relevance (1-5) - selection of important content from the source. The summary should include only important information from the source document.
Annotators were instructed to penalize summaries which contained redundancies and excess information.

Evaluation Steps:

1. Read the summary and the source document carefully.
2. Compare the summary to the source document and identify the main points of the article.
3. Assess how well the summary covers the main points of the article, and how much irrelevant or redundant information it contains.
4. Assign a relevance score from 1 to 5.


Example:


Source Text:

{{Document}}

Summary:

{{Summary}}


Evaluation Form (scores ONLY):

- Relevance:
```

Figure C.9: Prompt to evaluate the relevance of generated summaries using LLM

Appendix D

User Study Documents

D.1 Wireframe Diagrams

In this section, we show the wireframe diagrams for our user study.

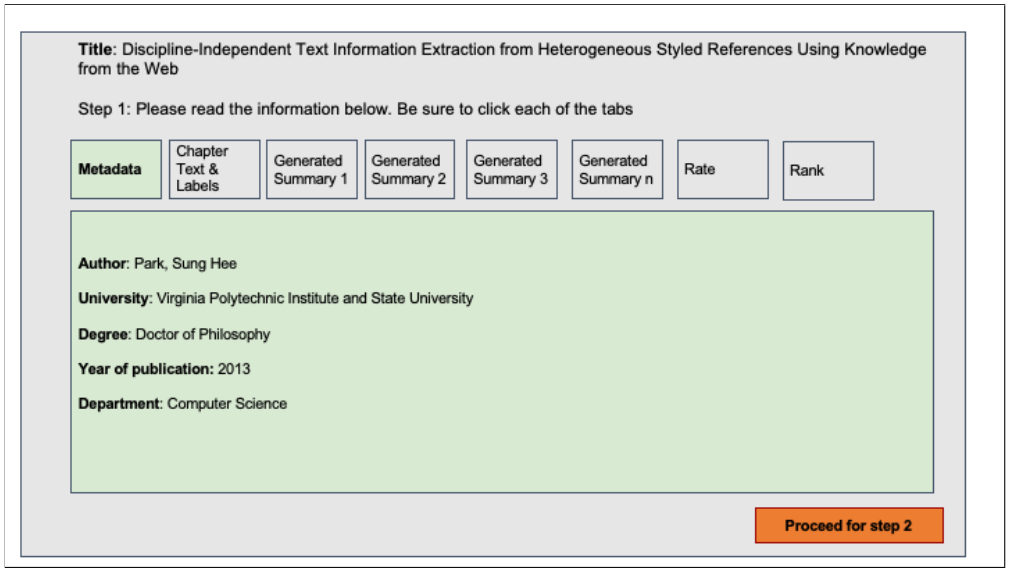


Figure D.1: Human evaluation framework wireframe - 1: Page displaying document metadata

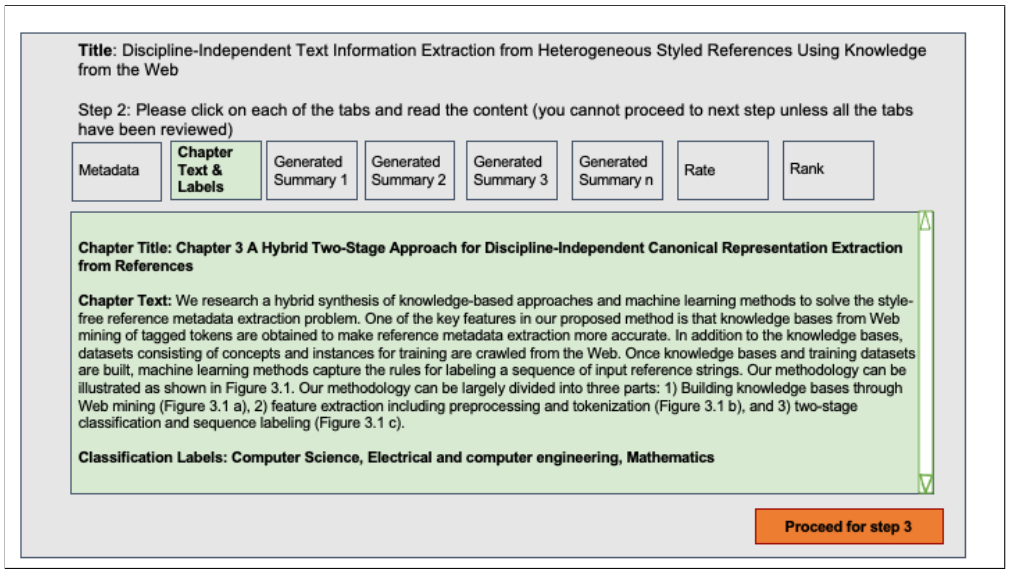


Figure D.2: Evaluation framework wireframe - 2: Page displaying chapter text

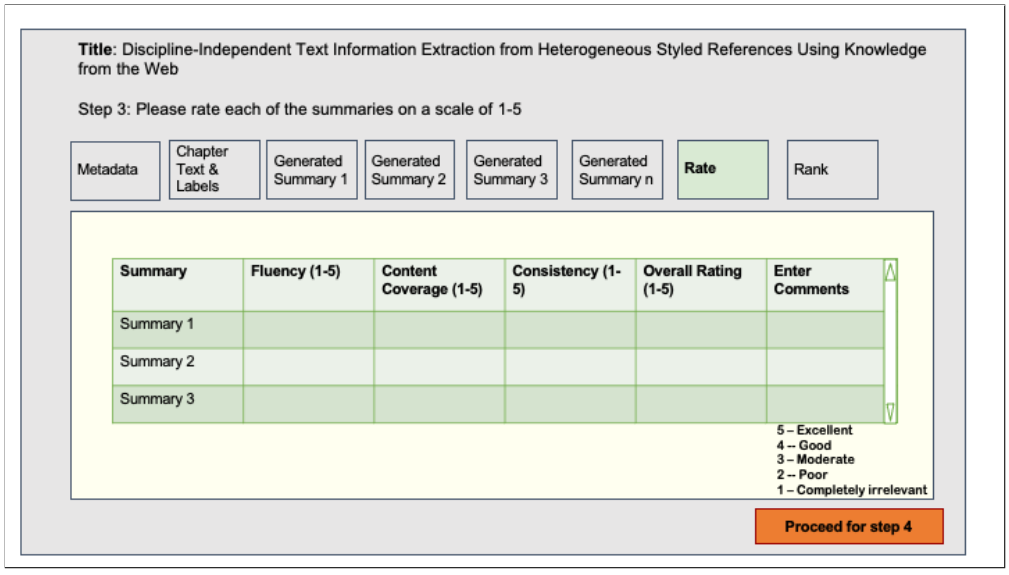


Figure D.3: Evaluation framework wireframe - 3: Page displaying summary ratings

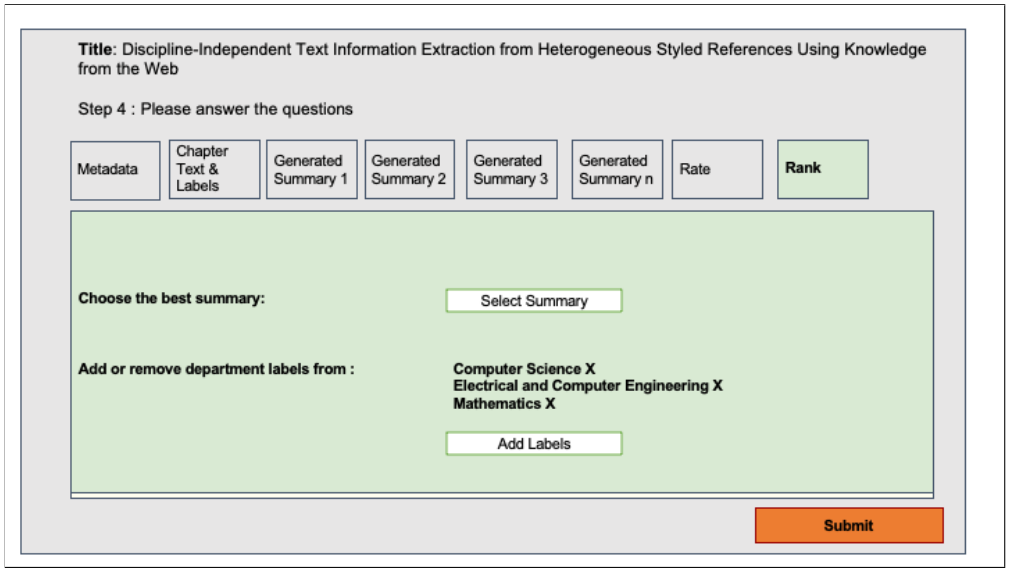


Figure D.4: Evaluation framework wireframe - 4: Page displaying classification

D.2 IRB Approval Letter

In this section, we add the IRB approval letter that enables us to recruit users to evaluate the research discussed in this document.

Division of Scholarly Integrity and Research Compliance
Institutional Review Board
North End Center, Suite 4120 (MC 0497)
300 Turner Street NW
Blacksburg, Virginia 24061
540/231-3732
irb@vt.edu
http://www.research.vt.edu/sirc/hrpp

MEMORANDUM

DATE: March 22, 2024

TO: Edward Fox, Bill Ingram, Harini Kandru, Bipasha Banerjee

FROM: Virginia Tech Institutional Review Board (FWA00000572)

PROTOCOL TITLE: Human evaluation of ETD chapter summaries and classification labels.

IRB NUMBER: **23-687**

Effective March 22, 2024, the Virginia Tech Human Research Protection Program (HRPP) determined that this protocol meets the criteria for exemption from IRB review under 45 CFR 46.104(d) category (ies) 2(i).

Ongoing IRB review and approval by this organization is not required. This determination applies only to the activities described in the IRB submission and does not apply should any changes be made. If changes are made and there are questions about whether these activities impact the exempt determination, please submit an amendment to the HRPP for a determination.

This exempt determination does not apply to any collaborating institution(s). The Virginia Tech HRPP and IRB cannot provide an exemption that overrides the jurisdiction of a local IRB or other institutional mechanism for determining exemptions.

All investigators (listed above) are required to comply with the researcher requirements outlined at:

https://secure.research.vt.edu/external/irb/responsibilities.htm

(Please review responsibilities before beginning your research.)

PROTOCOL INFORMATION:

Determined As:	**Exempt, under 45 CFR 46.104(d) category(ies) 2(i)**
Protocol Determination Date:	**July 6, 2023**

ASSOCIATED FUNDING:

The table on the following page indicates whether grant proposals are related to this protocol, and which of the listed proposals, if any, have been compared to this protocol, if required.

SPECIAL INSTRUCTIONS:

The protocol is being amended to add Harini Kandru to the research team. The information on the recruitment flyer has been rearranged.

Date*	OSP Number	Sponsor	Grant Comparison Conducted?
06/26/2023	PYQZV5V5	Institute of Museum & Library Services (Title: Opening Books and the National Corpus of Graduate Research)	Not required (Exempt approval)

* Date this proposal number was compared, assessed as not requiring comparison, or comparison information was revised.

If this protocol is to cover any other grant proposals, please contact the HRPP office (irb@vt.edu) immediately.

D.3 CS Pilot Study Assignments

In this section, we show the Canvas assignments that were distributed to CS4624 and CS5604 for the Computer Science user study.

Summarization Evaluation

Published | Edit

This assignment is like those for guest presentations, except the guest has prepared a short video and tutorial, so there is not something on the course calendar except a due date entry.

- -

Please look at the summarization tutorial at **https://virginiatech.questionpro.com/t/AYpWXZ1Mgf (https://virginiatech.questionpro.com/t/AYpWXZ1Mgf)** and view the video explaining the tutorial which is **in Canvas in Files (https://canvas.vt.edu/media_attachments_iframe/31952876?type=video&embedded=true)**. Please get in touch with Bipasha (**bipashabanerjee@vt.edu (mailto:bipashabanerjee@vt.edu)**) to get your personalized survey link and if you have any questions regarding the survey.
Using that survey instrument, you will be carrying out the following steps. If you fully and conscientiously complete those steps, you will get full credit for this assignment.
1. Read the ETD metadata.
2. Open a new window or tab in your browser for the chapter text PDF, and keep that open until the assignment is completed.
3. For each of 4 generated summaries:
3a. Read the summary.
3b. Assess the summary according to each of the criteria given on the rating page for that summary.
4. Rank the summaries, assigning each a rank, with rank 1 identifying the one that is best.
5. As needed, use the arrow key to go back, and the Next button to go forward, to navigate among the survey pages.
6. When assessment and ranking is complete, share any additional comments.
7. Enter Done when you are finished with all of the above.
8. For this assignment, enter into the text box your name and the date/time when you completed the steps above. Submit that in Canvas as your entry for this assignment.

Points	1
Submitting	a text entry box

Due	For	Available from	Until
13	Everyone	Jan 22 at 12am	Mar 14 at 11:59pm

+ Rubric

IR experimental pilot study option 1: Summarization evaluation

Published | Edit

The **Overview page (https://canvas.vt.edu/courses/176258/pages/f2023-overview)** 's section on Grading explains:

"5%: Participating in an IR experimental pilot study.
* There will be a series of these to choose from; exactly 1 should be carried out.
* These will be announced when ready, and available through Canvas.
* They will cover experiments about: summarization, topic modeling, etc."

You must complete exactly 1 of the pilot study options. Doing this assignment is one of those options. See details below.

- -

Please look at the summarization tutorial at **https://virginiatech.questionpro.com/t/AYpWXZz6jL (https://virginiatech.questionpro.com/t/AYpWXZz6jL)** .
If you are interested in this, please get in touch with Bipasha (**bipashabanerjee@vt.edu (mailto:bipashabanerjee@vt.edu)**) to get your personalized survey link.
Using that survey instrument, you will be carrying out the following steps. If you fully and conscientiously complete those steps, you will get full credit for this assignment.
1. Read the ETD metadata.
2. Open a new window or tab in your browser for the chapter text PDF, and keep that open until the assignment is completed.
3. For each of 4 generated summaries:
3a. Read the summary.
3b. Assess the summary according to each of the criteria given on the rating page for that summary.
4. Rank the summaries, assigning each a rank, with rank 1 identifying the one that is best.
5. As needed, use the arrow key to go back, and the Next button to go forward, to navigate among the survey pages.
6. When assessment and ranking is complete, share any additional comments.
7. Enter Done when you are finished with all of the above.
8. For this assignment, enter into the text box your name and the date/time when you completed the steps above. Submit that in Canvas as your entry for this assignment.

Points 5

Submitting a text entry box

D.4 Recruitment

This study is targeted towards VT faculty and graduate students. The following email was sent to our contacts across the University, spanning a wide variety of disciplines.

Dear Dr. [Professor name],

I am a Ph.D. candidate in computer science working under the guidance of Dr. Edward Fox.

My dissertation work relates to bringing computational access to Electronic Theses and Dissertations (ETDs) by providing users with machine-generated chapter summaries and classification labels. I am looking for participants among faculty and graduate students to evaluate ETD chapter summaries and classification labels from diverse disciplines.

I would be very grateful if you could broadcast this information to your department to get participants (VT faculty and graduate students) to engage in this study. Would you please forward my request to people in your department? Please let me know if that works, or kindly reply to this message if you have questions.

My study proposal (VT IRB-23-687) has been approved as IRB-exempt. The recruitment email, recruitment flier, and IRB approval letter are attached. Please let me know if you need more information from me.

Looking forward to hearing from you. Thank you so much for your time.

Regards,

Bipasha Banerjee

bipashabanerjee@vt.edu

Ph.D. Candidate

CS@VT

D.5 Participant Recruitment Email and Fliers

The following email and flier was distributed in the VT community through emails and listservs.

Recruitment Email:

Subject: Participants needed for Human Evaluation of ETD chapter summaries and classification labels.

Email body:

Hi Everyone,

We are seeking participants for evaluating ETD chapter summaries and classification labels (IRB #23-687). We are conducting a study to evaluate automatically generated chapter summary and chapter classification labels for electronic theses and dissertations (ETDs).

We are looking for research participants who meet the following criteria:

Current graduate student or faculty member at Virginia Tech Regularly read and use academic research papers. 18 years of age or older

Participants will need to complete the study in one sitting lasting approximately 30 minutes. The study takes place remotely. For more questions contact Bipasha Banerjee at bipashabanerjee@vt.edu

Regards,

Bipasha Banerjee

Ph.D. Candidate in Computer Science

Virginia Tech

Research Participants Needed – ETD chapter summary and chapter label evaluation.

We are conducting a study to evaluate automatically generated chapter summary and chapter classification labels for electronic theses and dissertations (ETDs) IRB #23-687.

The study involves an online/remote activity that should require approximately 30 minutes.
We are looking for research participants who meet the following criteria:

✓ Current graduate student or faculty member at Virginia Tech
✓ Regularly read and use academic research papers.
✓ 18 years of age or older

If interested contact:
bipashabanerjee@vt.edu

Project Investigators

Dr. Edward Fox – fox@vt.edu
Bill Ingram – waingram@vt.edu
Bipasha Banerjee – bipashabanerjee@vt.edu
Harini Kandru – harini@vt.edu

Appendix E

User Study Additional Results

We report the user study results for all non-CS departments aggregated together in Chapter 6, Section 6.4.3. We report all the individual results for non-CS disciplines in the VT wide user study in Figures E.1, E.2, E.3 and E.4 for a better understanding of how each of the disciplines performs.

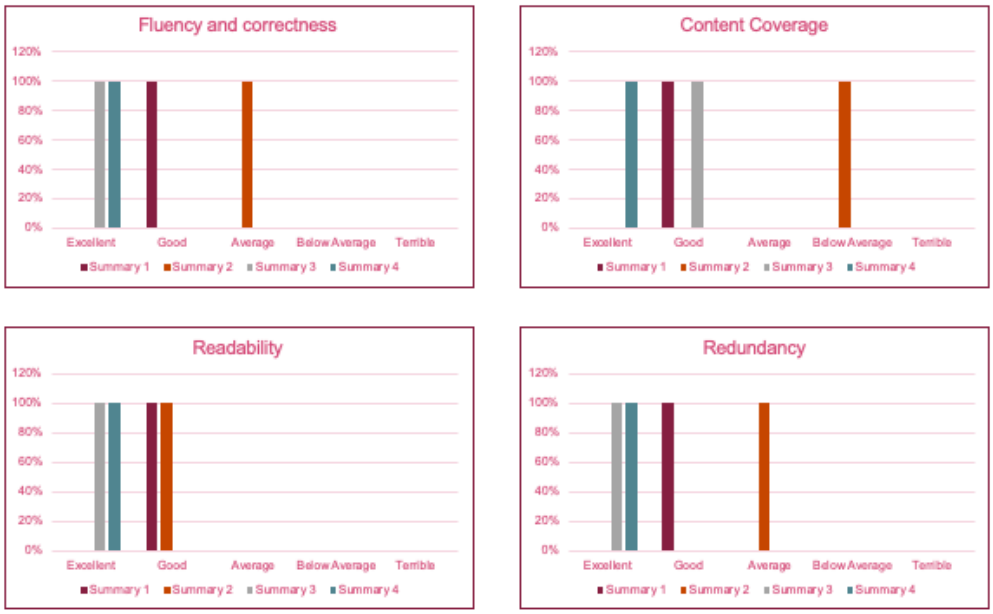


Figure E.1: Psychology user evaluation

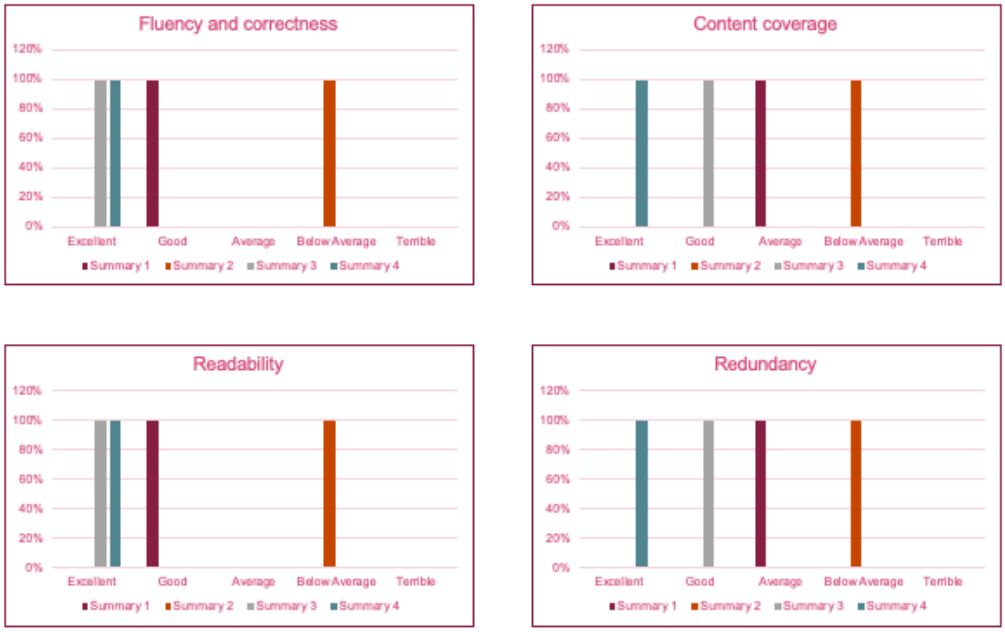


Figure E.2: Mechanical Engineering user evaluation

E.1 Weighted Average Calculation

We use a weighted average method to aggregate the user rating scores for each summary and each metric. Users in the study were asked to rate each summary on 4 metrics – "Fluency and correctness", "Consistency", "Readability", and "Redundancy". For each of the metrics, we had 5 scale ratings – "Excellent", "Good", "Average", "Below Average" and "Terrible". We use the assigned weights for each of the scale labels. For example, "Excellent" was given a weight of 5, and so on and so forth. We then calculate the weighted average based on the number of people rating each of the summaries. Scores from this calculation have been reported in Chapter 6, Tables 6.7, 6.9 and 6.15.

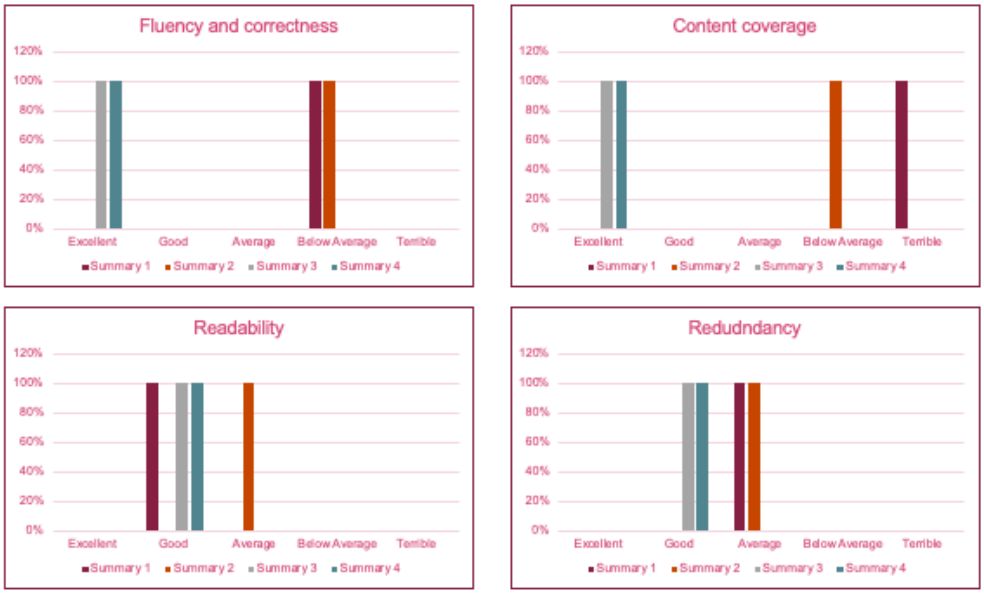


Figure E.3: Ecology user evaluation

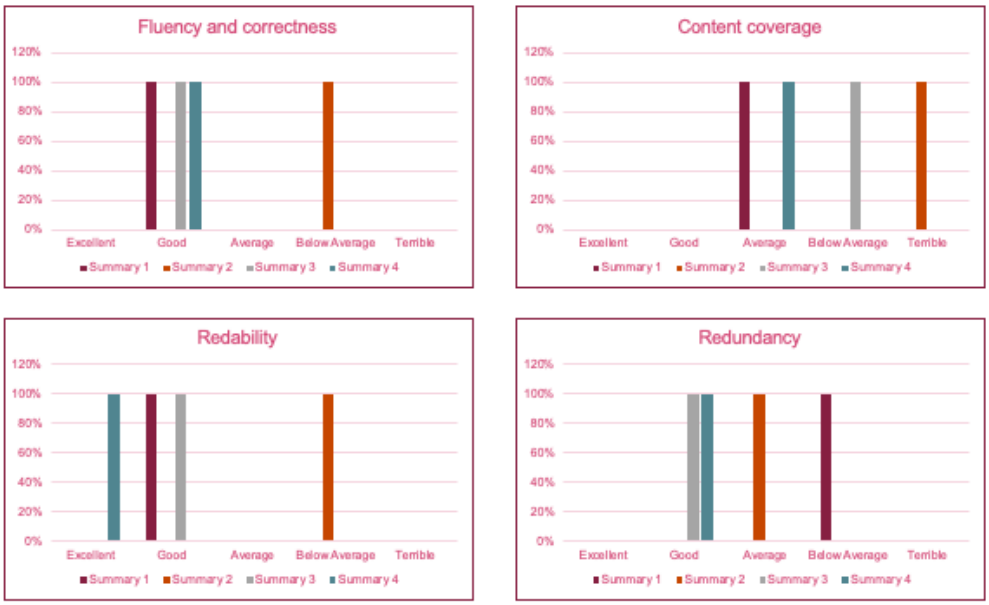


Figure E.4: Neuroscience user evaluation