Final Report
CS 5604: Information Storage and Retrieval

Team 4: Language Models, Classification, and Summarization

Kanad Naleshwarkar, Gayatri Bhatambarekar, Zeel Desai, Aishwarya Kumaran, Shadab Haque, Adithya Harish Srinivasan Manikandan

Instructed by Dr. Edward A. Fox

SMEs: Bipasha Banerjee, Sara Ahmadi

December 17, 2023

Virginia Polytechnic Institute and State University

Blacksburg, Virginia, 24061, USA

ABSTRACT

The CS5604 class at Virginia Tech has been tasked with developing an information retrieval and analysis system that can handle the collection of data of at least 500,000 Electronic Theses and Dissertations (ETDs), under the direction of Dr. Edward A. Fox. This program should function as a search engine with a variety of capabilities, including browsing, searching, giving suggestions, and rating search results. The class has been split into six teams to execute this job, and each team has been given a specific task. The goal of this report is to provide an overview of Team 4's contribution, which focuses on classification, summarization, and language models. Our prime tasks were testing out various models for classification and summarization. During the course of this project, we evaluated models developed by the previous team working on this task and explored various strategies to improve them. For the classification task, we fine-tuned the SciBERT model to get standardized subject category labels that are in accordance with ProQuest. We also evaluated a large language model, LLaMA 2, for the classification task, and after comparing its performance with the fine-tuned SciBERT model, we observed that LLaMA 2 was not efficient enough for a large-scale system that the class was working on. For summarization, we evaluated summaries generated by various transformer, non-transformer, and LLM-based models. The five models that we evaluated for summarization were TextRank, LexRank, LSA, BigBirdPegasus, and LLaMA 2 7B. We observed that although TextRank and BigBirdPegasus had comparable results, the summaries generated by TextRank were more comprehensive. This experimentation gave us valuable insight into the complexities of processing a large set of documents and performing tasks such as classification and summarization. Additionally, it allowed us to explore the deployment of these models in a production environment to evaluate their performance at scale.

Keywords: Summarization, Classification, LLMs, LLaMA, Subject Categories

Contents

List of Figures

List of Tables

List of Abbreviations

API	Application Programming Interface
BERT	Bidirectional Encoder Representations from Transformers
CI/CD	Continuous Integration Continuous Deployment
ETDs	Electronic Theses and Dissertations
GPT	Generative Pre-trained Transformer
LLaMA	Large Language Model Meta AI
LLM	Large Language Model
NLP	Natural Language Processing
POC	Proof of Concept
RNN	Recurrent Neural Network
SME	Subject Matter Expert

Chapter 1

Overview

1.1 Introduction

Team 4 has been entrusted with the responsibility of helping develop an Information Retrieval and Analysis system capable of efficiently and reliably handling a collection of over 500,000 ETDs. Our primary focus within this project revolves around leveraging advanced language models to enhance the segmentation, summarization, and classification of these ETDs.

To commence our work, we are closely collaborating with Team 5, responsible for overseeing database management and information flow within the system. This collaboration is crucial for acquiring the necessary data to fuel our research and development efforts.

Our initial objective is to summarize the segmented ETD chapters and then classify these summaries into various academic disciplines. In pursuit of this goal, we are dedicated to refining and optimizing existing summarization and classification models, ensuring they deliver superior performance. Our efforts encompass the exploration of diverse configurations for existing summarization models. This includes adjustments to parameters such as the maximum context length, choice of summarization models, resulting summary lengths, and strategies to mitigate repetition.

In parallel with these endeavors, we investigated the use of open-source large language models like LLaMA 2 [42] to support our capabilities in both classification and summarization tasks. We explored prompt-tuning methodologies for classification and summarization to get the inference before having to fine-tune these models, thereby trying to avoid computationally heavy workloads.

Ultimately, our goal is to integrate our models with the broader components of the retrieval system. This integration will facilitate the analysis of ETDs, enabling comprehensive and insightful examinations of this valuable academic content.

1.1.1 User Stories

Table 1.1 outlines user stories for a system handling ETDs. The first story, for experimenters, involves generating three summarized versions of a chapter text. The second story, also for experimenters, focuses on retrieving disciplines associated with a specific ETD. The third story, designed for curators, encompasses processing a batch of ETDs through summarization and classification pipelines, demonstrating a comprehensive approach to managing academic texts on a larger scale.

User Story	**Description**
User story 1	As an experimenter, I want to provide a chapter text and obtain 5 different summarized versions.
User story 2	As an experimenter, I want to fetch disciplines associated with an ETD.
User story 3	As a curator, I want to process a batch of ETDs and run through summarization and classification pipelines.

Table 1.1: User stories

1.2 Discussions with Subject Matter Experts

During our discussions with SMEs in the field of ETD Classification, Summarization, and Language Models, which included invaluable insights from Bipasha and Sara, we engaged in a comprehensive exploration of our project objectives and associated timelines. This dialogue was essential not only for aligning our project with the expectations of potential end users but also for gaining a deeper understanding of the dataset at hand.

Our initial task was to review and execute the existing code. However, it became clear that dataset curation for evaluating chapter summaries had not been completed, resulting in a shortage of ground truth data. Consequently, we were given the responsibility of creating a comprehensive evaluation dataset. This initiative aimed to enhance the robustness of our summarization pipeline.

For classification, the prior team utilized a set of labels that did not align with the ProQuest subject categories [35]. To overcome this, we were given an updated dataset with suitable labels. Our task now is to retrain the classification model so that it provides accurate subject categories.

Our discussions emphasized the critical importance of evaluating summarization techniques and exploring language models, with specific attention to LLaMA 2 [42].

We were responsible for improving current models and exploring new models with potentially superior outcomes. Ultimately, our task involved assessing the models we developed, including the LLaMA 2 model, to determine and finalize the one that surpasses all others in performance.

Chapter 2

Literature Review

2.1 Classification

ETDs are a valuable repository of academic knowledge, covering a wide range of subjects. With the growing volume of digital academic documents, automated document classification systems are increasingly important. These systems aid in efficient document indexing and retrieval and also contribute to academic research by uncovering emerging trends.

In recent years, the emergence of powerful pre-trained language models like BERT [12] and SciBERT [4] has significantly impacted document classification, leading to improved accuracy and efficiency. We drew inspiration from the work of the previous Fall 2022 class [14] while developing our models. In the upcoming sections, we will delve into the different models that we have explored.

2.1.1 BERT: Bidirectional Encoder Representations from Transformers

BERT [12], developed by Google AI, has emerged as a breakthrough in NLP due to its ability to capture contextual information bi-directionally. This pre-trained model has demonstrated success in various NLP tasks, including sentiment analysis, named entity recognition, and text classification. BERT's contextual embeddings have allowed it to understand the semantics and relationships between words and phrases, making it particularly suitable for document understanding and classification.

2.1.2 SciBERT: Domain-Specific Language Model

SciBERT [4], an extension of BERT [12], is specifically fine-tuned on a corpus of scientific literature. This specialization equips SciBERT with domain-specific knowledge and enhances its ability to comprehend scholarly documents. As a result, SciBERT [4] has emerged as a tool for classifying academic texts, including ETDs.

In the paper "Summarizing ETDs with deep learning" [20], the authors used the SciBERT [4] language model based on its promising performance in diverse academic disciplines. They created a dedicated pipeline for training and evaluating SciBERT, and notably, SciBERT [4] outperformed BERT on their evaluation dataset. This approach highlights the benefits of using domain-specific language models in academic document classification.

2.2 Summarization

One of the most extensively studied topics in the NLP domain is summarization, which is to compress the document's length while maintaining its content. There are two types of summarization approaches, namely extractive and abstractive, depending on whether particular phrases are taken into account as they exist in the original text or whether new sentences are produced using NLP techniques [15].

Adhikari et al. [38] explore a wide range of machine learning models, including recurrent neural networks, convolutional neural networks, and transformer-based architectures, to generate abstractive and extractive summaries. They discuss the use of attention mechanisms, reinforcement learning, and reinforcement-based methods for improving summary quality. Additionally, the paper covers diverse datasets, evaluation metrics, and applications across domains. Overall, they provide insights into the evolving landscape of NLP-based machine-learning approaches for text summarization.

"A Scalable Summarization System Using Robust NLP" [2] has presented a summarization system designed to efficiently generate summaries from large volumes of text data while ensuring robustness through NLP techniques. The paper discusses the development of a system capable of summarizing extensive textual content effectively.

"Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond" [29] explores the use of sequence-to-sequence RNNs and their extensions for the task of abstractive text summarization. The paper discusses the foundational approach of using RNNs to generate summaries by encoding source text and decoding summaries in a sequence-to-sequence fashion. It goes beyond the basic framework by introducing improvements such as attention mechanisms, pointer-generator networks, and coverage mechanisms to enhance the quality of generated summaries. The paper also highlights the challenges and future directions in abstractive text summarization, making it a valuable resource for researchers in the field.

"Summarizing ETDs with deep learning" [20] explores the innovative use of ETDs for text summarization. The study assesses deep learning summarization techniques on a large ETD collection, addressing challenges posed by domain-specific language and diverse subjects. To tackle data constraints, the research introduces transfer learning for ETD chapter summarization and evaluates multiple deep learning models. It provides insights into adapting summarization to specific domains, and underscores the potential of transfer learning in handling data limitations.

TextRank [28] is a graph-based ranking model for text analysis, which identifies key sentences and extracts keywords. It constructs a sentence graph, linking sentences based on content overlap measured by shared words, and uses the PageRank [32] algorithm to select the most significant sentences for summarization. For keyword extraction, TextRank creates a word network by linking consecutive words, assigning higher weights to frequent pairs, and then applying PageRank to select the top one-third as relevant keywords. The model recommends parts of speech tagging or a custom dictionary to filter out common stopwords, and focus on meaningful content

LexRank, implemented by Güneş Erkan et al. [13], is an algorithm used in natural language processing and text summarization. It is designed to automatically extract key sentences or phrases from a text document to create a summary, and it is based on the concept of graph-based ranking. LexRank was developed as an extension of the PageRank algorithm used by Google to rank web pages in search results. LexRank has been used in various applications, including text summarization, document clustering, and information retrieval.

Latent Semantic Analysis (LSA) [31] is a natural language processing technique that can be used in text summarization, among other applications. LSA is primarily a dimensionality reduction method that helps uncover the hidden relationships between words and concepts in a large collection of texts. It operates on the principle that words with similar meanings tend to occur in similar contexts.

The BigBirdPegasus model was introduced by Zaheer et al. [45]. “Big Bird: Transformers for Longer Sequences” extends Transformer-based models like BERT to process longer sequences using a sparse-attention mechanism. The model combines sparse attention, global attention, and random attention, enabling it to approximate full attention for long documents. BigBirdPegasus has demonstrated improved performance on various long-document NLP tasks, including question answering and summarization, compared to models like BERT [12] or RoBERTa [25].

The Longformer model [3] is designed to work with long texts more efficiently than traditional Transformer models, which struggle with lengthy documents due to increased computational requirements. Longformer addresses this issue by adjusting how it focuses on different parts of the text, primarily considering nearby words and certain keywords regardless of their position. This change significantly reduces the amount of computation needed, enabling Longformer to handle longer texts with ease. In practical applications, Longformer consistently outperforms models like RoBERTa in tasks involving long documents and achieves impressive results on benchmarks like text8 [27] and enwik8 [26], which test a model’s predictive capabilities.

2.3 Language Models

Large Language Models (LLMs) leverage a class of deep learning algorithms known as transformers [43], comprising self-attention, multi-head self-attention, feed-forward layers, and normalization. Transformers process sequences of tokens (words, characters, or subword units) from training data. Different LLMs adopt varied training strategies. For instance, BERT [12] is trained in a supervised manner to predict masked items in a sequence, whereas GPT [36] focuses on predicting the next token.

The training involves pretraining the model through unsupervised learning on a large text corpus, followed by fine-tuning on specific tasks like summarization and translation. The foundational architecture of LLMs involves transformers [43], which have superseded previous models like LSTMs [17] and GRUs [6] due to their parallel processing capabilities. Key components of this architecture include an attention mechanism, which assigns significance to different tokens, and multi-head attention, allowing simultaneous focus on various parts of the input.

OpenAI's series of models - GPT-1.0 [36], GPT-2.0 [37], GPT-3.0 [5], and GPT-4.0 [30] - have been pivotal in LLM advancements, demonstrating exceptional ability in generating human-like text. Each version exhibits notable improvements in architecture, training corpus, and functionalities, including zero-shot learning, few-shot learning, and many-shot learning.

Google's PaLM [8] and PaLM 2 [1] models, built on the transformer architecture, introduced modifications like SwiGLU [39] activation, parallel layers, multi-query attention, and RoPE embeddings [40]. These models emphasize efficiency, training speed, and scalability. PaLM 2, in particular, employs compute optimal scaling for enhanced performance.

Meta's LLaMA [41] and its successor, LLaMA 2 [42], also follow the transformer architecture, integrating advancements like SwiGLU [39], rotational position embeddings, and root mean squared normalization. They focus on reduced computational demands and enhanced efficiency, with a pretraining corpus extending to 2 trillion tokens.

The paper titled "Improved Sampling Techniques for Learning an Imbalanced Data Set" by Lauron and Pabico [23], addresses the challenges posed by imbalanced datasets in classification tasks. They introduce five new sampling techniques, including SMOTERandRep, Lax Random Oversampling (LRO), and Lax Random Undersampling (LRU), proposing improvements over traditional methods like Random Undersampling, Random Oversampling, and SMOTE. Evaluation metrics such as F-measure and G-mean are employed to assess classifier performance, revealing that, without sampling techniques, satisfactory performance is observed only for the majority class. Notably, SMOTE, SMOTERandRep, and LRO consistently outperform other techniques, showcasing their effectiveness in enhancing classifier performance, with LRO achieving the highest G-mean value. The paper underscores the importance of sampling methods in handling imbalanced datasets and advocates for further exploration, particularly on larger multiclass imbalanced datasets.

Chapter 3

Requirements

3.1 Overall Project Requirements

The goal of our project was to develop an information retrieval system that would process a collection of 500k ETDs and provide a way for users to find and interact with ETDs. Some of the specific tasks included:

- Developing a UI so that users can sign up and interact with the system.
- Indexing the available ETDs.
- Building a recommendation system for users.
- Building an object-detection system to extract text from scanned ETDs.
- Hosting the system on Kubernetes containers and ensuring uptime.

3.2 Team 4 Requirements

Team 4 was tasked with experimenting with classification and summarization models for assigning subject categories and generating summaries for the ETDs uploaded by users. This process included assessing open-source large language models, such as LLaMA 2, to compare their performance with traditional summarization and classification models. Once the models were optimized, our responsibility extended to developing utility codes for getting inferences for the collection of 500k ETDs. These inferences, i.e., summaries and subject category tags, would help organize the collection of ETDs.

The utility code incorporated Kafka topics to manage data flow within the classification and summarization pipelines. It involves processes like retrieving ETD data from databases through APIs, converting the data into batches, processing each batch for model inference, and storing the results back into the database.

Both summarization and classification pipelines were developed to handle the scale required by the information retrieval system being developed in the class.

3.3 Collaborations

Collaboration with other Teams:

- Team 4 was provided ETDs segmented into chapters for the summarization process.
- The classification tags generated by our classification model will be utilized by Team 2 to build a search and recommendation system.
- Team 5 was responsible for handling the infrastructure to host our classification and summarization models.

Chapter 4

Design

4.1 Classification

Figure 4.1 depicts the overall flow of the classification task.

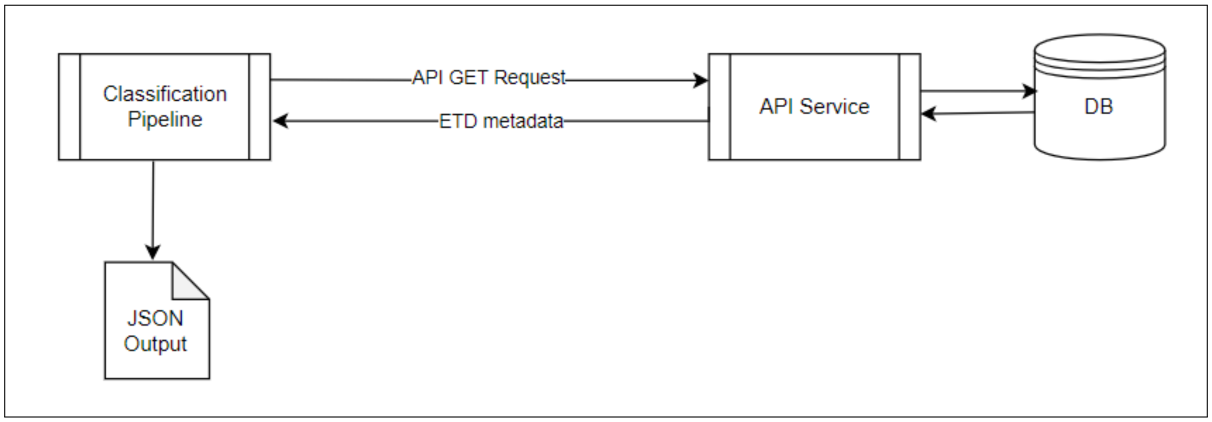


Figure 4.1: Classification overall flow

For classification, the ETD title-abstract pairs are collected from the database and through the APIs provided by Team 5. After fetching this data, it is processed in batches to get inferences. Each title-abstract pair is classified into either of the 18 ProQuest level-2 subject categories [35]. Along with the most probable subject category, the output of the classification model also contains the second-most probable subject category.

Our approach to ETD classification is influenced by the work of the Fall 2022 class [14]. We have chosen to utilize the fine-tuned SciBERT [4] model for classifying summarized ETD chapters.

However, we have made a change in our approach while training our model. In our implementation, we use the ProQuest level 2 subject categories [35] as labels. These 21 subject categories cover various disciplines. This categorization strikes a balance between being detailed enough to be meaningful and broad enough to avoid excessive specificity, making it well-suited for classifying ETDs.

Table 4.1 depicts the label distribution in the training dataset (provided by the SME) after preprocessing. The training dataset is discussed in detail in Section 5.2.1.

Label	**Count**
ENGINEERING	10430
EDUCATION	5565
MATHEMATICAL AND PHYSICAL SCIENCES	4690
SOCIAL SCIENCES	3501
FINE AND PERFORMING ARTS	2719
BEHAVIORAL SCIENCES	2276
LANGUAGE AND LITERATURE	1885
ENGINEERING | ARCHITECTURE | ENVIRONMENTAL SCIENCES	1797
GEOSCIENCES	1372
HISTORY	1083
ARCHITECTURE	809
BIOLOGICAL SCIENCES	722
COMMUNICATIONS AND INFORMATION SCIENCES	698
AGRICULTURE	672
BUSINESS	669
HEALTH AND MEDICAL SCIENCES | EDUCATION	294
COMMUNICATIONS AND INFORMATION SCIENCES | INTERDISCIPLINARY	258
PHILOSOPHY AND RELIGION	209
BEHAVIORAL SCIENCES | EDUCATION	203
Total	39852

Table 4.1: Distribution of labels in the classification dataset

4.2 Summarization

4.2.1 Workflow of the Model

Summarizing ETDs is a challenging job as these documents are typically very long, and there's a shortage of appropriate training data. Consequently, our proposed methodology is centered around chapter-level summarization, as depicted in Figure 4.2.

Every chapter is subjected to preprocessing and tokenization of the text. Real-world data generally contains noise, and missing values, and may be in an unusable format that cannot be directly used for machine learning models. Data preprocessing is a required task for cleaning the data and making it suitable for a machine learning model, which also increases the accuracy and efficiency. Tokenization is the first step in preprocessing text data. It involves dividing a text document into smaller units known as tokens, which can be words, phrases, or even individual characters. These tokens are subsequently transformed into numerical representations. After tokenization, each word or phrase is converted into a dense vector of real numbers. The significant advantage of these embeddings is their ability to capture semantic meanings and relationships between words or phrases, which enables machines to understand and process human language efficiently.

The numerical vector that represents that text is subsequently forwarded to the text summarizer model which in turn produces a summary of the entire chapter. By iterating through these steps for each chapter input, we obtain a collection of chapter summaries. These summaries can be employed individually to address the specific needs of the end user.

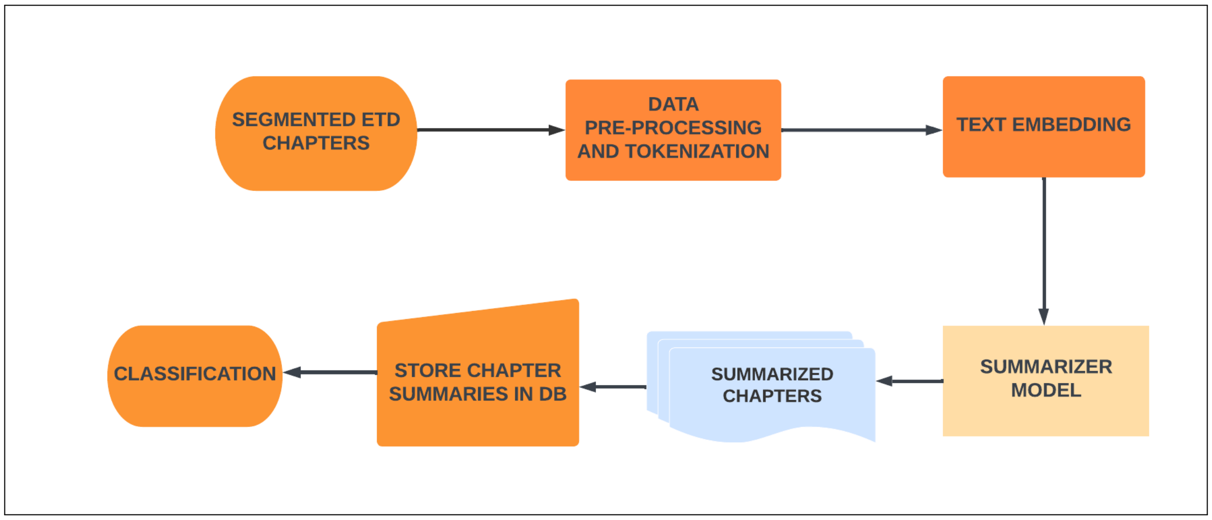


Figure 4.2: ETD chapter summarization flow

4.3 Language Models

4.3.1 Summarization

The goal is to use LLaMA 2 [42] to get summaries of the ETDs. Since LLaMA 2 is available in three model sizes, i.e., 7, 13, and 70 billion parameters, training all the parameters for our task would be computationally expensive and out of scope for this project. The most common way to use LLaMA 2 for a custom task would be to instruction-tune already pretrained models trained upon a huge dataset suitable for the summarization task. To instruction-tune LLaMA 2 for ETD summarization the arXiv Summarization Dataset [9] and BookSum dataset [21] have been selected.

arXiv: This dataset was designed for the task of summarizing long documents, particularly scientific papers, which are often characterized by complex structures and detailed discourse. The choice to utilize this particular dataset for fine-tuning a summarization model is because of its domain-specificity. For the ETDs that include scientific content, fine-tuning this data allows the model to become more attuned to the domain-specific language and terminologies of research papers, which is crucial for producing accurate and relevant summaries. The dataset has also been empirically validated by the original authors.

BookSum: BookSum is an extensive dataset assembly designed for long-form narrative summarization, encompassing a range of literary works like novels, plays, and stories. It features expertly crafted summaries at varying levels of detail, including paragraph, chapter, and book length. The selection of this specific dataset is pivotal for the task at hand. In the case of Electronic Theses and Dissertations (ETDs) that encompass non-scientific subjects, it's crucial to employ a specialized dataset tailored for summarization. Disciplines like History, Education, and Behavioral Sciences exhibit unique content structures and thematic elements, necessitating a dataset that is fine-tuned to effectively summarize ETDs within these fields.

However, before proceeding with fine-tuning, both of the datasets needed to be preprocessed. The arXiv Summarization dataset, while a valuable resource, contains various forms of noise that need to be addressed for optimal use. This includes the presence of LaTeX notations, special characters, and symbols, which are commonplace in academic papers but can disrupt the fine-tuning process. Additionally, the reference sections in these documents, although essential for academic purposes, may introduce irrelevant information for certain computational tasks. To enhance the dataset's utility, it's important to carefully filter out these elements. The BookSum dataset is characterized by its relatively low levels of noise, primarily because it consists of straightforward book chapters. This attribute enhances its usability for fine-tuning, as it minimizes the need for extensive data cleaning or preprocessing typically associated with more complex or technical datasets such as the arXiv Summarization Dataset.

In preparing the arXiv Summarization Dataset for analysis, we have employed a series of functions that utilize regular expressions (regex) for efficient preprocessing:

1. **clean_latex**: Removes LaTeX notations, including command sequences, subscript and superscript expressions, and brackets from the text, enhancing readability and processing.

2. **remove_citations**: Targets and removes citation markers like `@xcite` and `@xmath0`, decluttering the text from academic placeholders.

3. **remove_xmath_instances**: Eliminates instances of mathematical notation markers, focusing the text on verbal content.

4. **remove_empty_parentheses**: Searches for and removes empty parentheses, further cleaning the text.

5. **remove_references**: Extracts and removes reference entries, including authors, publication years, and DOIs, to focus on the core content of the text.

6. **remove_special_characters_and_digits**: Strips away all non-alphabetic characters and digits, leaving only letters and spaces for text analysis purposes.

7. **remove_extra_spaces**: Reduces multiple spaces to a single space and trims leading and trailing spaces for text uniformity.

8. **remove_specific_word**: Allows for the removal of a specific word from the text, useful for filtering out unwanted terms.

9. **remove_references1**: Similar to 'remove_references' but uses a more comprehensive regex pattern to remove a wider variety of reference formats for thorough academic citation cleanup.

In our project, we faced a constraint with the LLaMA 2 model's maximum token limit of 4096, a challenge particularly evident when dealing with Electronic Theses and Dissertations

(ETDs) that typically exceed this length. Although we initially considered utilizing the variant of LLaMA 2 capable of handling a larger context of 32,000 tokens, this option was not feasible due to its substantial memory requirements, which exceeded our available resources. Consequently, our approach was constrained to using the standard LLaMA 2 model, limited to processing 4096 tokens at a time. This limitation likely resulted in partial content analysis, a reduction in contextual depth, and a skew towards prioritizing initial segments of the content.

Further, to prepare the dataset for instruction-tuning, an instruction "**Below is a text followed by its abstract. Write a summary of the article based on the abstract provided**" is used for every row of the arXiv Summarization dataset. This methodology was similarly applied to the BookSum dataset, employing the same structured approach to enhance the model's capability in synthesizing and contextualizing information from longer narrative forms, such as books. The 'Text' column, which contains a prompt followed by the article and its abstract, is central to instruction-tuning. It provides the model with a clear task (summarization) within a specific context (using the abstract to summarize the article). This format teaches the model to correlate detailed content with its summarized form. The prompt explicitly instructs the model on the task, reinforcing its understanding of the summarization process and how to approach it. By feeding the 'Text' column to the language model, we are effectively training it to generate summaries based on abstracts. During training, the model learns to parse the article and abstract, and then apply the instruction to generate a summary. This enhances its ability to follow complex instructions and produce relevant outputs. In our approach, we treat each book chapter as the equivalent of an article, and its corresponding summary is viewed as the abstract. This analogy allows for a consistent framework in processing and analyzing both of the datasets.

The approach of providing a language model like LLaMA 2 with both a text and a summary is because summaries often focus on the high-level findings and conclusions of a paper. During training, the model has been instruction-tuned with a structured approach where the ‘Text’ column includes a prompt (“Below is a text followed by its abstract. Write a summary of the article based on the abstract provided”), the article, and its abstract. This is a classic case of instruction-tuning, where the model learns to generate summaries based on the specific instructions provided in the prompt. The model learned to understand the structure and content of both the article and its abstract, and how to condense the article into a summary similar to the abstract. In the testing phase, we are providing only the article without the abstract. This represents a different task for the model compared to its training. The model now has to generate a summary without the direct guidance of an abstract. It relies on its understanding of the article’s content, structure, and the inherent skill of summarization it developed during training. This approach tests the model’s ability to generalize its learning. It assesses whether the model can apply its summarization skills without relying on the structure it was trained on (i.e., summarizing with an abstract as a reference). However, no experimentation was done with the training set. The instruction-tuned model was used to generate summaries on the test set as shown in Table 4.2. This approach was developed following the guidance and recommendations of SMEs.

Table 4.2 depicts the overall structure of the dataset.

Feature	**Description**
features	[“article”, “abstract”, “text”]
num_rows	6398

Table 4.2: Overview of the arXiv dataset

Since LLMs including LLaMA 2 require heavy computational resources for performing optimally, quantization is used for fine-tuning. Quantization is a method that streamlines numerical data by decreasing its precision level. Within the realm of LLMs, this process entails transitioning from high-accuracy floating-point figures to more simplified, lower-accuracy fixed-point formats. Quantization markedly diminishes the memory requirements by shrinking the model parameters, thereby facilitating the deployment of LLMs on platforms with limited resources.

QLoRA [11] is also used to assist with the efficient memory usage. QLoRA optimizes the process by introducing minimal trainable parameters, known as adapters, across each LLM layer. This strategy allows for the original parameters to remain unchanged, with only the lightweight adapters undergoing modifications during fine-tuning. This selective alteration significantly reduces memory usage, making the procedure more manageable.

4.3.2 Classification

The initial stage involved loading the dataset from the CSV file named `'Classification_training.csv'`. To focus on relevant information, the DataFrame was filtered to include only the "abstract" and "Label" columns, thus removing any extraneous data. The loaded dataset comprised 'abstract' and 'Label' features and contained 39,852 rows, each corresponding to a different document.

To facilitate the classification task, a specific prompt format was created for each entry in the dataset. This format included an introduction stating the task **"Below is an abstract of an article. Categorize it appropriately."**, the abstract of the article, the category label. This structured format was applied to each data entry to standardize the input for the classification model.

The dataset was shuffled and split into training, validation, and test sets using an 80-10-10 split ratio. The training, validation, and test sets were then combined into a final DatasetDict. This final dataset was organized into distinct "train", "validation", and "test" partitions, each containing data entries in the standardized prompt format.

For the intital run, the classification task utilized the `'meta-llama/Llama-2-7b-hf'` model. The model was configured with BitsAndBytesConfig to enable 4-bit quantization. The compute dtype was set to `torch.bfloat16`. The tokenizer for the model was set with the end-of-sequence token as the padding token and padding side set to “right”. The model was prepared for k-bit training to enhance efficiency with limited resources. LoRA [18] configuration was applied to adapt the model for efficient learning, with settings like lora_alpha = **16**, lora_dropout = **0.1**, and rank = **64**. A higher lora_alpha means the adjustments can be more significant, potentially leading to more substantial changes in the model’s behavior during fine-tuning. This can be useful when you want the model to adapt more aggressively to new tasks or data. Lora_dropout means that a certain percent of the neurons in the LoRA layers are randomly dropped out. This helps in making the fine-tuned model more robust and less likely to overfit to the training data. Rank means that the low-rank matrices are relatively small compared to the original model’s weights, which helps in keeping the additional computational cost low. The choice of rank is a balance between the effectiveness of adaptation and the additional computational cost and memory usage. LoRA configuration was then integrated into the model to enhance its adaptability. After training, the model was loaded and LoRA layers were merged with the base model, finalizing it for evaluation and deployment.

Chapter 5

Implementation

5.1 Tools

In our project, we have strategically selected a set of tools and technologies to ensure efficient development, seamless collaboration, and the ability to harness state-of-the-art methods in NLP and Deep Learning/Machine Learning. Here's an overview of the key tools and technologies that will be instrumental in our project's success:

1. Python: Python [16] serves as our primary programming language due to its versatility and robust support for various NLP and Deep Learning libraries. Python's ease of use allows us to rapidly prototype and explore different solutions, making it the ideal choice for our research-oriented project.

2. Docker: Docker [10] plays a crucial role in providing a well-defined and isolated environment for running our specific solutions. It encapsulates all necessary prerequisites, libraries, and system-level binaries required for seamless functioning. We will leverage Docker to create containers for each of our project's services, including summarization, segmentation, and classification. This ensures consistency and portability across different environments.

3. GitLab: GitLab [7] serves as our code repository, facilitating effective version control and collaboration among team members. It enables the maintenance of various feature branches and supports the implementation of CI/CD pipelines for streamlined deployment processes.

4. PyTorch: PyTorch [19] is our machine learning library of choice, offering a wide range of tools and capabilities for building layers, models, and data pipelines. PyTorch accelerates development by providing a flexible and intuitive framework while also supporting hardware-accelerated training, essential for deep learning tasks.

5. HuggingFace: HuggingFace [44] is a valuable library built around PyTorch, specializing in state-of-the-art transformer models. It simplifies access to pre-trained weights for transformer architectures like BERT, enabling us to experiment with cutting-edge models efficiently. This library greatly enhances our ability to work with advanced language models and embeddings.

6. LangChain: LangChain [22] is a specialized library that simplifies the creation and usage of large language models. It streamlines the process of working with extensive language models, making it easier to integrate them into our project.

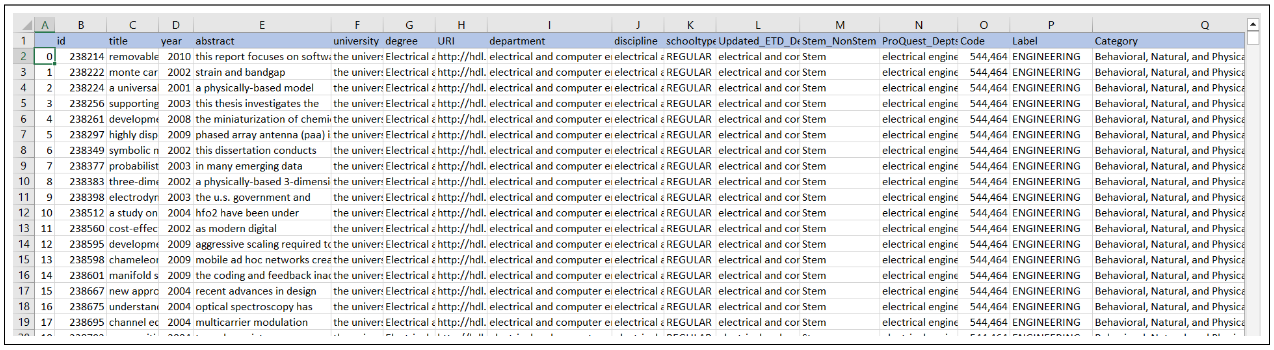
7. Trello: Trello [34] serves as our project management tool, allowing us to assign tasks, track progress, and ensure efficient coordination among team members. Its user-friendly interface supports agile project management practices, helping us stay organized and on track.

5.2 Classification

5.2.1 Classification Fine-tuning

To align the model output with the ProQuest subject categories [35], we have fine-tuned the classification model again with an updated label set. To accomplish this, we are utilizing a dataset provided by our SME. This dataset leverages ETD abstracts as features and employs ProQuest subject categories across all three levels as labels. Specifically, we have chosen to focus on the ProQuest subject categories at level 2 for our fine-tuning efforts.

Figure 5.1 shows a snapshot of the training data that we are using:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1		id	title	year	abstract	university	degree	URI	department	discipline	schooltype	Updated_ETD_De	Stem_NonStem	ProQuest_Depts	Code	Label	Category
2	0	238214	removable	2010	this report focuses on softwa	the univers	Electrical a	http://hdl.	electrical and computer e	electrical a	REGULAR	electrical and cor	Stem	electrical engine	544,464	ENGINEERING	Behavioral, Natural, and Physica
3	1	238222	monte car	2002	strain and bandgap	the univers	Electrical a	http://hdl.	electrical and computer e	electrical a	REGULAR	electrical and cor	Stem	electrical engine	544,464	ENGINEERING	Behavioral, Natural, and Physica
4	2	238224	a universal	2001	a physically-based model	the univers	Electrical a	http://hdl.	electrical and computer e	electrical a	REGULAR	electrical and cor	Stem	electrical engine	544,464	ENGINEERING	Behavioral, Natural, and Physica
5	3	238256	supporting	2003	this thesis investigates the	the univers	Electrical a	http://hdl.	electrical and computer e	electrical a	REGULAR	electrical and cor	Stem	electrical engine	544,464	ENGINEERING	Behavioral, Natural, and Physica
6	4	238261	developme	2008	the miniaturization of chemi	the univers	Electrical a	http://hdl.	electrical and computer e	electrical a	REGULAR	electrical and cor	Stem	electrical engine	544,464	ENGINEERING	Behavioral, Natural, and Physica
7	5	238297	highly disp	2009	phased array antenna (paa) i	the univers	Electrical a	http://hdl.	electrical and computer e	electrical a	REGULAR	electrical and cor	Stem	electrical engine	544,464	ENGINEERING	Behavioral, Natural, and Physica
8	6	238349	symbolic n	2002	this dissertation conducts	the univers	Electrical a	http://hdl.	electrical and computer e	electrical a	REGULAR	electrical and cor	Stem	electrical engine	544,464	ENGINEERING	Behavioral, Natural, and Physica
9	7	238377	probabilist	2003	in many emerging data	the univers	Electrical a	http://hdl.	electrical and computer e	electrical a	REGULAR	electrical and cor	Stem	electrical engine	544,464	ENGINEERING	Behavioral, Natural, and Physica
10	8	238383	three-dime	2002	a physically-based 3-dimensi	the univers	Electrical a	http://hdl.	electrical and computer e	electrical a	REGULAR	electrical and cor	Stem	electrical engine	544,464	ENGINEERING	Behavioral, Natural, and Physica
11	9	238398	electrodyn	2003	the u.s. government and	the univers	Electrical a	http://hdl.	electrical and computer e	electrical a	REGULAR	electrical and cor	Stem	electrical engine	544,464	ENGINEERING	Behavioral, Natural, and Physica
12	10	238512	a study on	2004	hfo2 have been under	the univers	Electrical a	http://hdl.	electrical and computer e	electrical a	REGULAR	electrical and cor	Stem	electrical engine	544,464	ENGINEERING	Behavioral, Natural, and Physica
13	11	238560	cost-effec	2002	as modern digital	the univers	Electrical a	http://hdl.	electrical and computer e	electrical a	REGULAR	electrical and cor	Stem	electrical engine	544,464	ENGINEERING	Behavioral, Natural, and Physica
14	12	238595	developme	2009	aggressive scaling required to	the univers	Electrical a	http://hdl.	electrical and computer e	electrical a	REGULAR	electrical and cor	Stem	electrical engine	544,464	ENGINEERING	Behavioral, Natural, and Physica
15	13	238598	chameleor	2009	mobile ad hoc networks crea	the univers	Electrical a	http://hdl.	electrical and computer e	electrical a	REGULAR	electrical and cor	Stem	electrical engine	544,464	ENGINEERING	Behavioral, Natural, and Physica
16	14	238601	manifold s	2009	the coding and feedback inac	the univers	Electrical a	http://hdl.	electrical and computer e	electrical a	REGULAR	electrical and cor	Stem	electrical engine	544,464	ENGINEERING	Behavioral, Natural, and Physica
17	15	238667	new appro	2004	recent advances in design	the univers	Electrical a	http://hdl.	electrical and computer e	electrical a	REGULAR	electrical and cor	Stem	electrical engine	544,464	ENGINEERING	Behavioral, Natural, and Physica
18	16	238675	understan	2004	optical spectroscopy has	the univers	Electrical a	http://hdl.	electrical and computer e	electrical a	REGULAR	electrical and cor	Stem	electrical engine	544,464	ENGINEERING	Behavioral, Natural, and Physica
19	17	238695	channel ec	2004	multicarrier modulation	the univers	Electrical a	http://hdl.	electrical and computer e	electrical a	REGULAR	electrical and cor	Stem	electrical engine	544,464	ENGINEERING	Behavioral, Natural, and Physica

Figure 5.1: Classification training dataset snapshot

Before training the model, we are conducting pre-processing steps to resolve various issues found within the dataset.

In the dataset, which includes roughly 39,000 entries, we found that about 2,600 entries were missing their abstracts. These entries have been removed.

Moreover, we observed that for some abstracts, labels were a combination of two or more subject categories separated by a pipe symbol "|". For example, "ENGINEERING | ARCHITECTURE | ENVIRONMENTAL SCIENCES". To address this issue, we split such labels to get individual subject categories and applied one-hot encoding on the label set. This approach will help us get more granular output from the model as opposed to the label powerset approach followed by the previous implementation by the Fall 2022 class.

To address the uneven distribution of labels, we have assigned weights to them based on their frequency — the rarer the label, the higher its weight. These weights are then factored into the loss function to balance the influence of each label during training.

5.2.2 Classification Fine-tuning Results

After processing the data, to improve the SciBERT model's performance on our test data, we experimented with different settings, trying various batch sizes, learning rates, and numbers of training epochs. We considered the following parameters and values to train the model:

- Batch sizes: 16, 32, 64
- Learning rates: 2×10^{-4}, 1×10^{-5}
- Epochs: 15, 20

The best-performing model, with batch size of 64, learning rate of 1×10^{-5}, and that is trained for 20 epochs, yielded an F1 score of 0.74 and a log loss of 0.72 on the test set, both averaged over all batches in the test set. These values meet our expectations and provide us with a reliable classification model.

5.2.3 Classification Fine-tuning Statistics

The model fine-tuning took approximately 2.5 hours to complete. After the training was complete we worked on the helper file to run inference on the actual ETDs present in the database created by Team 5. We calculated the time taken for our inference pipeline and observed that to fetch 1000 title-abstract pairs it took around 73.7 sec and to run inference on these 1000 title abstract pairs it took another 2.4 sec These numbers were helpful for comparison with the classification task we were working on using LLaMA 2.

5.3 Summarization

5.3.1 Summarization Results

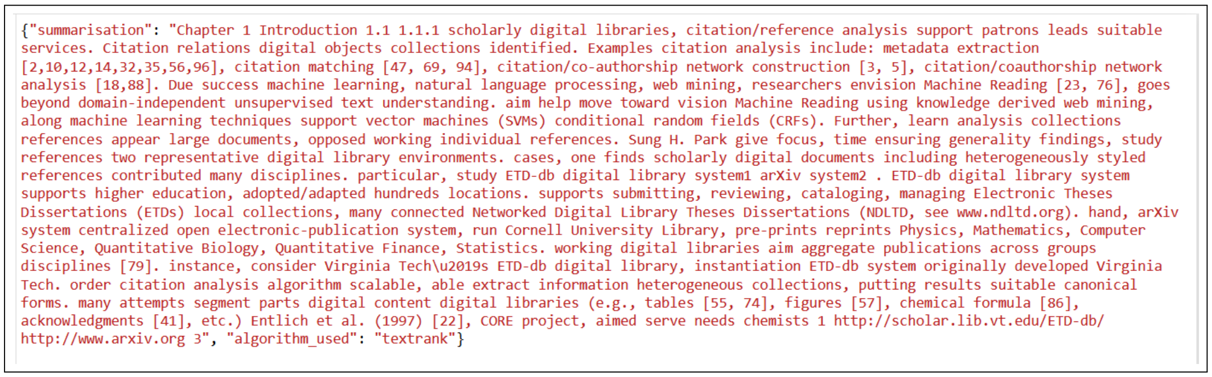
We have utilized four different algorithms to generate summaries for segmented chapters. The first three algorithms – TextRank, LexRank, and LSA – are non-transformer models. In addition, we have implemented the BigBirdPegasus model, which is a transformer-based algorithm.

Non-transformer models, like TextRank, LexRank, and LSA, rely on traditional approaches such as statistical analysis, graph-based methods, and singular value decomposition, respectively. These models tend to be simpler and faster, but may not capture the complexities and nuances of language as effectively as transformer models.

On the other hand, transformer models, such as BigBirdPegasus, leverage attention mechanisms to process input sequences. This enables the model to focus on different parts of the input text and capture long-range dependencies, leading to more context-aware and coherent outputs. While transformer models are generally more powerful and capable of producing better results, they are also computationally intensive and can be slower to train and infer compared to non-transformer models.

We employed the chapter "Discipline-Independent Text Information Extraction from Heterogeneously Styled References Using Knowledge from the Web" [33] for all four algorithms in order to assess their performance. The output generated was a summarized text of approximately 500 words.

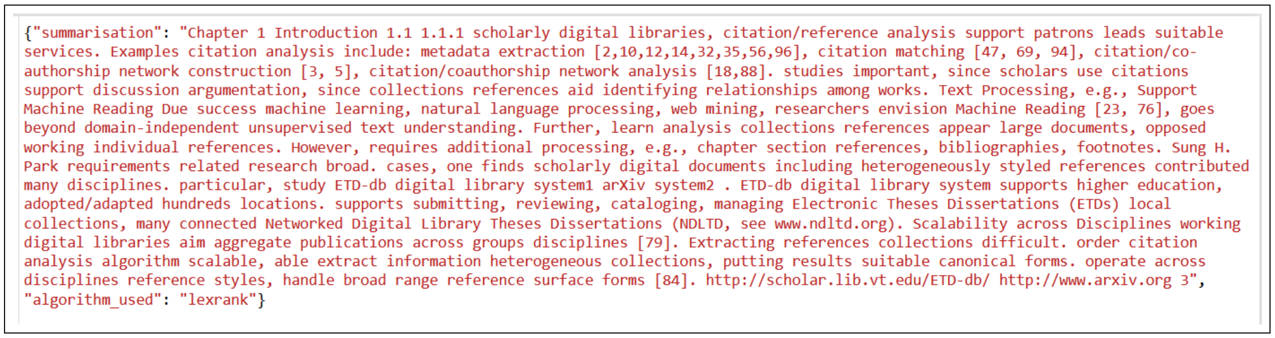
The TextRank [28] model was implemented using the TextRankSummarizer from the Sumy library. Sumy is a Python library that employs different algorithms for text summarization tasks. Figure 5.2 illustrates the summarized results of the chapter.



```
{"summarisation": "Chapter 1 Introduction 1.1 1.1.1 scholarly digital libraries, citation/reference analysis support patrons leads suitable
services. Citation relations digital objects collections identified. Examples citation analysis include: metadata extraction
[2,10,12,14,32,35,56,96], citation matching [47, 69, 94], citation/co-authorship network construction [3, 5], citation/coauthorship network
analysis [18,88]. Due success machine learning, natural language processing, web mining, researchers envision Machine Reading [23, 76], goes
beyond domain-independent unsupervised text understanding. aim help move toward vision Machine Reading using knowledge derived web mining,
along machine learning techniques support vector machines (SVMs) conditional random fields (CRFs). Further, learn analysis collections
references appear large documents, opposed working individual references. Sung H. Park give focus, time ensuring generality findings, study
references two representative digital library environments. cases, one finds scholarly digital documents including heterogeneously styled
references contributed many disciplines. particular, study ETD-db digital library system1 arXiv system2 . ETD-db digital library system
supports higher education, adopted/adapted hundreds locations. supports submitting, reviewing, cataloging, managing Electronic Theses
Dissertations (ETDs) local collections, many connected Networked Digital Library Theses Dissertations (NDLTD, see www.ndltd.org). hand, arXiv
system centralized open electronic-publication system, run Cornell University Library, pre-prints reprints Physics, Mathematics, Computer
Science, Quantitative Biology, Quantitative Finance, Statistics. working digital libraries aim aggregate publications across groups
disciplines [79]. instance, consider Virginia Tech\u2019s ETD-db digital library, instantiation ETD-db system originally developed Virginia
Tech. order citation analysis algorithm scalable, able extract information heterogeneous collections, putting results suitable canonical
forms. many attempts segment parts digital content digital libraries (e.g., tables [55, 74], figures [57], chemical formula [86],
acknowledgments [41], etc.) Entlich et al. (1997) [22], CORE project, aimed serve needs chemists 1 http://scholar.lib.vt.edu/ETD-db/
http://www.arxiv.org 3", "algorithm_used": "textrank"}
```

Figure 5.2: Summary generated using TextRank

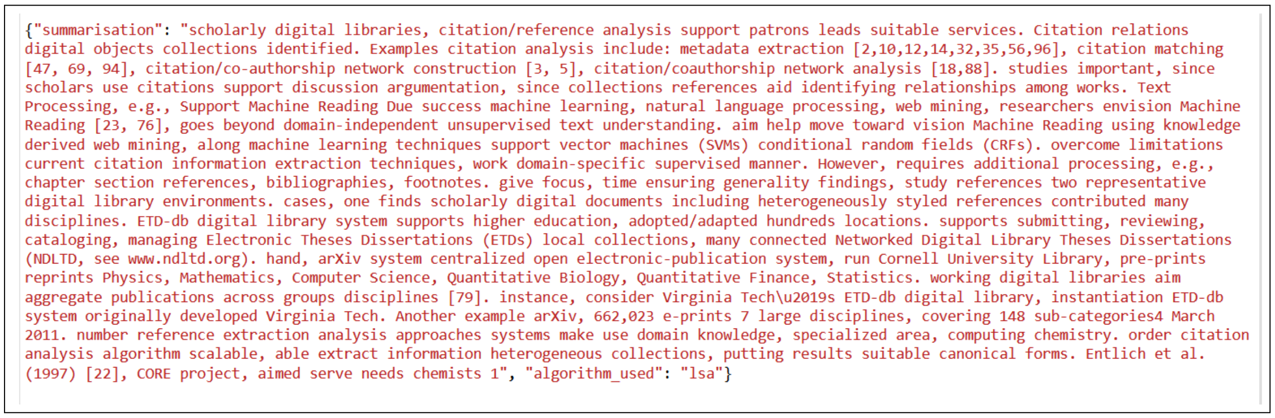
The LexRank [13] method is similar to TextRank and is based on the PageRank method. It uses a graph-based technique for automatic text summarization. The LexRankSummarizer model is employed for this task. Figure 5.3 illustrates the summarized chapter results using LexRank.



```
{"summarisation": "Chapter 1 Introduction 1.1 1.1.1 scholarly digital libraries, citation/reference analysis support patrons leads suitable
services. Examples citation analysis include: metadata extraction [2,10,12,14,32,35,56,96], citation matching [47, 69, 94], citation/co-
authorship network construction [3, 5], citation/coauthorship network analysis [18,88]. studies important, since scholars use citations
support discussion argumentation, since collections references aid identifying relationships among works. Text Processing, e.g., Support
Machine Reading Due success machine learning, natural language processing, web mining, researchers envision Machine Reading [23, 76], goes
beyond domain-independent unsupervised text understanding. Further, learn analysis collections references appear large documents, opposed
working individual references. However, requires additional processing, e.g., chapter section references, bibliographies, footnotes. Sung H.
Park requirements related research broad. cases, one finds scholarly digital documents including heterogeneously styled references contributed
many disciplines. particular, study ETD-db digital library system1 arXiv system2 . ETD-db digital library system supports higher education,
adopted/adapted hundreds locations. supports submitting, reviewing, cataloging, managing Electronic Theses Dissertations (ETDs) local
collections, many connected Networked Digital Library Theses Dissertations (NDLTD, see www.ndltd.org). Scalability across Disciplines working
digital libraries aim aggregate publications across groups disciplines [79]. Extracting references collections difficult. order citation
analysis algorithm scalable, able extract information heterogeneous collections, putting results suitable canonical forms. operate across
disciplines reference styles, handle broad range reference surface forms [84]. http://scholar.lib.vt.edu/ETD-db/ http://www.arxiv.org 3",
"algorithm_used": "lexrank"}
```

Figure 5.3: Summary generated using LexRank

Figure 5.4 illustrates the summarized results of the chapter using the LSA [31] model.



```
{"summarisation": "scholarly digital libraries, citation/reference analysis support patrons leads suitable services. Citation relations
digital objects collections identified. Examples citation analysis include: metadata extraction [2,10,12,14,32,35,56,96], citation matching
[47, 69, 94], citation/co-authorship network construction [3, 5], citation/coauthorship network analysis [18,88]. studies important, since
scholars use citations support discussion argumentation, since collections references aid identifying relationships among works. Text
Processing, e.g., Support Machine Reading Due success machine learning, natural language processing, web mining, researchers envision Machine
Reading [23, 76], goes beyond domain-independent unsupervised text understanding. aim help move toward vision Machine Reading using knowledge
derived web mining, along machine learning techniques support vector machines (SVMs) conditional random fields (CRFs). overcome limitations
current citation information extraction techniques, work domain-specific supervised manner. However, requires additional processing, e.g.,
chapter section references, bibliographies, footnotes. give focus, time ensuring generality findings, study references two representative
digital library environments. cases, one finds scholarly digital documents including heterogeneously styled references contributed many
disciplines. ETD-db digital library system supports higher education, adopted/adapted hundreds locations. supports submitting, reviewing,
cataloging, managing Electronic Theses Dissertations (ETDs) local collections, many connected Networked Digital Library Theses Dissertations
(NDLTD, see www.ndltd.org). hand, arXiv system centralized open electronic-publication system, run Cornell University Library, pre-prints
reprints Physics, Mathematics, Computer Science, Quantitative Biology, Quantitative Finance, Statistics. working digital libraries aim
aggregate publications across groups disciplines [79]. instance, consider Virginia Tech\u2019s ETD-db digital library, instantiation ETD-db
system originally developed Virginia Tech. Another example arXiv, 662,023 e-prints 7 large disciplines, covering 148 sub-categories4 March
2011. number reference extraction analysis approaches systems make use domain knowledge, specialized area, computing chemistry. order citation
analysis algorithm scalable, able extract information heterogeneous collections, putting results suitable canonical forms. Entlich et al.
(1997) [22], CORE project, aimed serve needs chemists 1", "algorithm_used": "lsa"}
```

Figure 5.4: Summary generated using LSA

The summarize transformers function is designed to generate a chapter summary specifically using the BigBirdPegasus model [45], a transformer-based model from Google. The function initializes variables for the summarization model and the summary results.

Within the function, the BigBirdPegasus model is selected based on the input configuration. If “BigBirdPegasus” is specified, the function uses Google’s implementation of this model.

The function takes the text from a chapter as input. It then uses the BigBirdPegasus model to generate a summary. Several parameters were configured to control our summarization pipeline:

- max_length: Sets the maximum acceptable length for the generated summary, ensuring it does not exceed 500 tokens.

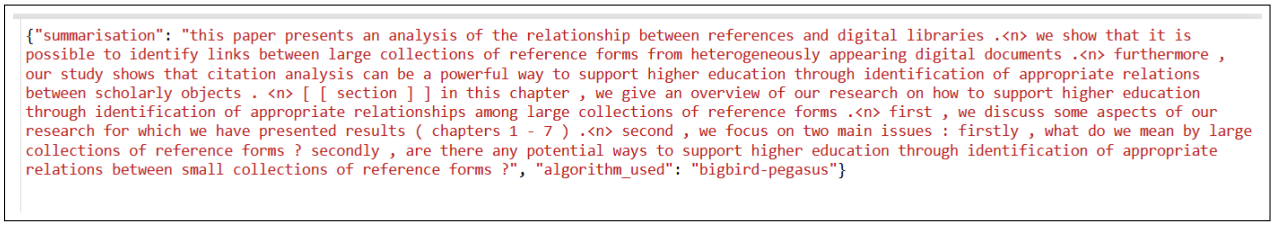
- min_length: Sets the minimum acceptable length for the summary, preventing it from being too short by requiring a minimum of 200 tokens.

- do_sample: Determines whether the summarization process involves random sampling. In this case, it is set to False, indicating a deterministic process without random variations.

- early_stopping: Enables early stopping during summarization, meaning the process concludes once specific criteria, such as achieving a certain level of confidence in the summary, are met.

Once the summary is generated, the function processes the output to ensure it is in string format. It also truncates the summary if it exceeds the maximum allowed output length. Finally, the processed summary is stored in the chapter summary attribute of the utility object.

The summary produced by the BigBirdPegasus model is then displayed in Figure 5.5.



```
{"summarisation": "this paper presents an analysis of the relationship between references and digital libraries .<n> we show that it is
possible to identify links between large collections of reference forms from heterogeneously appearing digital documents .<n> furthermore ,
our study shows that citation analysis can be a powerful way to support higher education through identification of appropriate relations
between scholarly objects . <n> [ [ section ] ] in this chapter , we give an overview of our research on how to support higher education
through identification of appropriate relationships among large collections of reference forms .<n> first , we discuss some aspects of our
research for which we have presented results ( chapters 1 - 7 ) .<n> second , we focus on two main issues : firstly , what do we mean by large
collections of reference forms ? secondly , are there any potential ways to support higher education through identification of appropriate
relations between small collections of reference forms ?", "algorithm_used": "bigbird-pegasus"}
```

Figure 5.5: Summary generated using BigBirdPegasus

After looking at the summaries generated by different models we found that:

- TextRank demonstrated simplicity in implementation, and we found that it is a lightweight and computationally efficient option as it excelled in quickly summarizing the content. Moreover, its algorithmic simplicity enhances transparency, allowing one to comprehend and customize the summarization process easily. However, TextRank struggled with capturing intricate contextual nuances, leading to potential oversimplification of complex content.

- BigBirdPegasus, a transformer-based architecture, stood out in its ability to grasp intricate contextual relationships, resulting in superior summarization quality. Its strength lies in understanding the semantics and context of the input text comprehensively. While it offered high-quality summaries, the downside was that it took a longer time to generate summaries.

- On the other hand, LSA and LexRank present some challenges. LSA, despite its focus on semantic relationships, struggled to capture fine-grained details due to its reliance on statistical patterns. LexRank, utilizing a graph-based approach, strikes a balance between simplicity and efficacy, but it faced challenges when dealing with content that lacked clear semantic structures.

5.3.2 Summarization Evaluation

Evaluation is a crucial aspect of our work, as it allows us to assess and confirm the quality, relevance, and effectiveness of the generated summaries by our models. The summary needs to be well structured without unnecessary stop words, repetitions, and unnecessary elaboration. There are two kinds of possible summaries, namely extractive and abstractive. Extractive summarization extracts the important sentences or keywords grouped together in order to form a concise summary, whereas abstractive summarization utilizes natural language techniques to interpret and understand aspects of the entire text, so it can generate new text.

In the evaluation of our text summarization models, we employed ROUGE metrics [24]. ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation. These metrics, including ROUGE-1, ROUGE-2, and ROUGE-L, offer a comprehensive assessment of summarization quality by measuring precision, recall, and F-measure. Here are the key aspects of ROUGE:

- **ROUGE-1 Score:** This metric evaluates the overlap of unigrams (individual words) between the generated text and the reference text. It's a measure of how many words are in common in both texts, giving an indication of the content's accuracy. ROUGE-1 is calculated by counting the number of overlapping unigrams in the generated summary and the reference summary, then calculating the recall (proportion of reference unigrams captured by the generated summary) and precision (proportion of generated unigrams present in the reference summary). The final score is often represented as an F1-score, which combines precision and recall.

- **ROUGE-2 Score:** This metric focuses on the overlap of bigrams (pairs of consecutive words) between the generated text and the reference text. It provides insights into how well the generated text captures two-word phrases from the reference text, which is important for assessing the coherence and flow of the summary. Similar to ROUGE-1, ROUGE-2 is calculated by identifying the common bigrams in both texts and computing the recall, precision, and F1-score. This metric is more stringent than ROUGE-1 as it requires matching pairs of words in the same order.

- **ROUGE-L Score:** ROUGE-L stands for ROUGE-Longest Common Subsequence. It evaluates the longest common subsequence of words between the generated and reference texts. Unlike ROUGE-1 and ROUGE-2, ROUGE-L does not require consecutive words to match but focuses on the order of words. It considers the longest sequence of words that appears in both the generated and reference texts, regardless of whether they are consecutive or not. It allows for non-consecutive matching of words in the sequence, emphasizing the overall order of words rather than requiring them to be adjacent to each other. This metric is particularly useful for assessing the sentence-level structure and fluency of the summary. The score is calculated based on the length of the longest common subsequence, with adjustments for recall and precision. ROUGE-L is sensitive to longer sequences of correct word order, making it a good indicator of overall summary quality, especially in terms of readability and coherence.

Table 5.1, Table 5.2, and Table 5.3 provide insights into how well the generated summaries align with reference summaries, considering both content and structure of one ETD. We have taken the reference summaries that were generated manually.

Model	**precision**	**recall**	**F1 score**
TextRank	0.258	0.319	0.285
LexRank	0.220	0.340	0.267
LSA	0.23	0.38	0.29
BigBirdPegasus	0.273	0.206	0.235

.

Table 5.1: Evaluation of summarization models - ROUGE-1

Model	**precision**	**recall**	**F1 score**
TextRank	0.083	0.1036	0.092
LexRank	0.067	0.103	0.081
LSA	0.6	0.1	0.08
BigBirdPegasus	0.034	0.025	0.029

.

Table 5.2: Evaluation of summarization models - ROUGE-2

Model	**precision**	**recall**	**F1 score**
TextRank	0.141	0.175	0.156
LexRank	0.117	0.180	0.141
LSA	0.11	0.18	0.14
BigBirdPegasus	0.164	0.123	0.141

.

Table 5.3: Evaluation of summarization models - ROUGE-L

In the context of ETD summarization tasks, achieving higher ROUGE scores indicates that the generated summaries effectively capture the essential information present in the reference summaries. After looking at the above tables we can see that the BigBirdPegasus model performs better followed by TextRank.

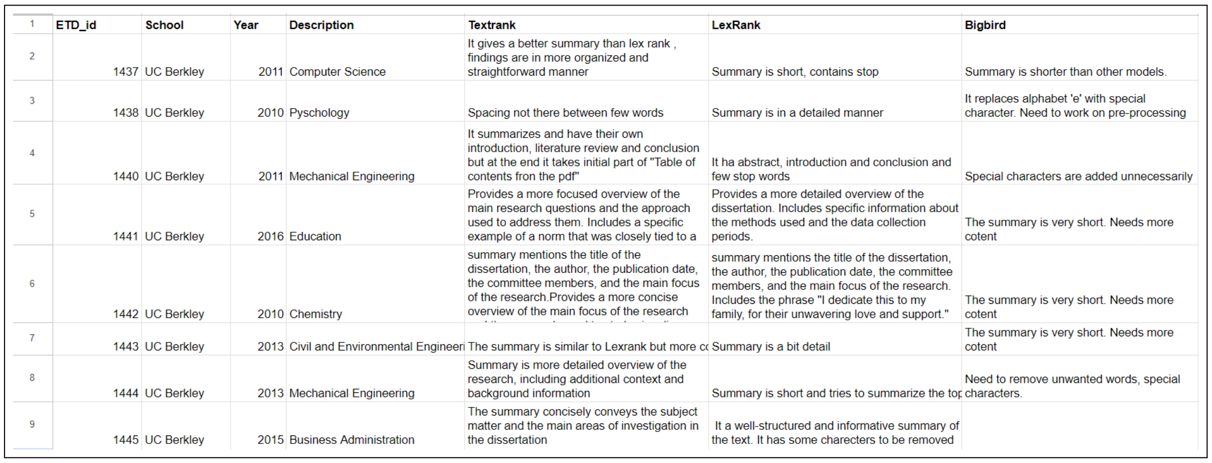
Table 5.4 presents the ROUGE-1 values of the summaries in comparison to the reference summary, which, in this case, was the abstract of the chapter.

Model	**precision**	**recall**	**F1 score**
TextRank	0.76	0.45	0.56
LexRank	0.74	0.39	0.51
LSA	0.70	0.56	0.46
BigBirdPegasus	0.55	0.08	0.1

.

Table 5.4: Evaluation of summaries with abstract - ROUGE-1

To complement the quantitative evaluation, we conducted a manual review of 8 ETDs as shown in Figure 5.6. This qualitative assessment revealed that TextRank outperformed other algorithms and also produced more appealing and quickly generated summaries. As a result of this comprehensive evaluation, TextRank emerged as the preferred choice for summarizing ETDs after facing strong competition from the BigbirdPegasus model.



1	ETD_id	School	Year	Description	Textrank	LexRank	Bigbird
2	1437	UC Berkley	2011	Computer Science	It gives a better summary than lex rank , findings are in more organized and straightforward manner	Summary is short, contains stop	Summary is shorter than other models.
3	1438	UC Berkley	2010	Pyschology	Spacing not there between few words	Summary is in a detailed manner	It replaces alphabet 'e' with special character. Need to work on pre-processing
4	1440	UC Berkley	2011	Mechanical Engineering	It summarizes and have their own introduction, literature review and conclusion but at the end it takes initial part of "Table of contents fron the pdf"	It ha abstract, introduction and conclusion and few stop words	Special characters are added unnecessarily
5	1441	UC Berkley	2016	Education	Provides a more focused overview of the main research questions and the approach used to address them. Includes a specific example of a norm that was closely tied to a	Provides a more detailed overview of the dissertation. Includes specific information about the methods used and the data collection periods.	The summary is very short. Needs more cotent
6	1442	UC Berkley	2010	Chemistry	summary mentions the title of the dissertation, the author, the publication date, the committee members, and the main focus of the research.Provides a more concise overview of the main focus of the research	summary mentions the title of the dissertation, the author, the publication date, the committee members, and the main focus of the research. Includes the phrase "I dedicate this to my family, for their unwavering love and support."	The summary is very short. Needs more cotent
7	1443	UC Berkley	2013	Civil and Environmental Engineeri	The summary is similar to Lexrank but more co	Summary is a bit detail	The summary is very short. Needs more cotent
8	1444	UC Berkley	2013	Mechanical Engineering	Summary is more detailed overview of the research, including additional context and background information	Summary is short and tries to summarize the top	Need to remove unwanted words, special characters.
9	1445	UC Berkley	2015	Business Administration	The summary concisely conveys the subject matter and the main areas of investigation in the dissertation	It a well-structured and informative summary of the text. It has some charecters to be removed	

Figure 5.6: Manual review of 8 ETDs

5.4 Experimenting with Large Language Models

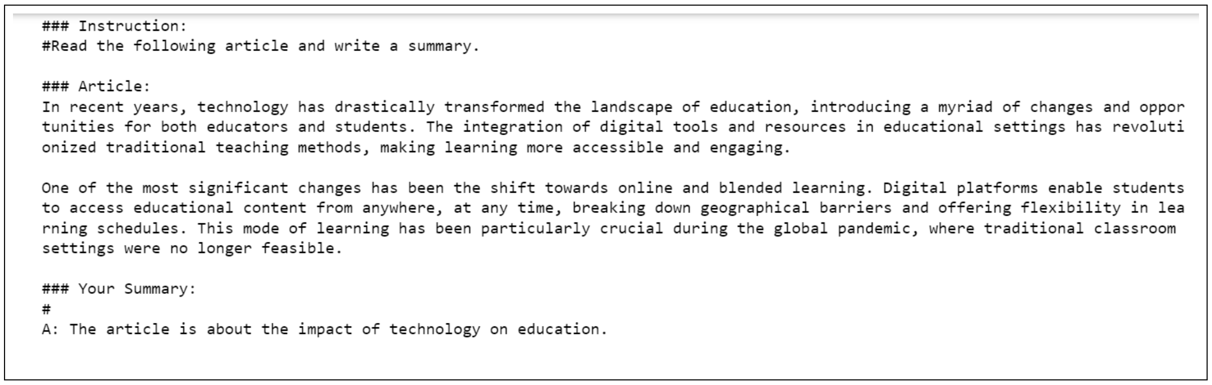
5.4.1 Summarization

Table 5.5 lists LLaMA 2 models and datasets used for experimentation.

Model	Dataset fine-tuned on
LLaMA-2-7B-32k	arXiv
LLaMA-2-7B	arXiv + BookSum
LLaMA-2-13B	arXiv + BookSum

Table 5.5: LLaMA 2 models and datasets used for experiments

The instruction-tuning of the `'togethercomputer/LLaMA-2-7B-32K'` model on the arXiv-summarization dataset aimed to generate summaries from the "article" column. It should be noted that despite using the LLaMA-2-7B-32K, the maximum token chosen was 4096, which is the same as LLaMA-7B. Also, no few-shot learning was done for any of the models. Inference using the base model is shown in Figure 5.7.



```
### Instruction:
#Read the following article and write a summary.

### Article:
In recent years, technology has drastically transformed the landscape of education, introducing a myriad of changes and oppor
tunities for both educators and students. The integration of digital tools and resources in educational settings has revoluti
onized traditional teaching methods, making learning more accessible and engaging.

One of the most significant changes has been the shift towards online and blended learning. Digital platforms enable students
to access educational content from anywhere, at any time, breaking down geographical barriers and offering flexibility in lea
rning schedules. This mode of learning has been particularly crucial during the global pandemic, where traditional classroom
settings were no longer feasible.

### Your Summary:
#
A: The article is about the impact of technology on education.
```

Figure 5.7: Summarized text of a paragraph

The inference results from the base model on arXiv dataset indicate a lack of improvement across various examples, with certain instances yielding no summary generation at all, while others produce summaries that are incoherent or nonsensical. This maybe be due to how the input data is being preprocessed and inputted for inference. Sometimes the inclusion of digits and special characters significantly contributes to the lack of coherence in the content.

The instruction-tuning process on arXiv dataset was configured with the following training arguments:

- `num_train_epochs`: 2
- `per_device_train_batch_size`: 6 if using flash attention, otherwise 2
- `gradient_accumulation_steps`: 2
- `gradient_checkpointing`: True
- `optim`: “paged_adamw_32bit”
- `logging_steps`: 10
- `save_strategy`: “epoch”
- `learning_rate`: 3e-5
- `fp16`: True
- `max_grad_norm`: 0.3
- `warmup_ratio`: 0.03
- `lr_scheduler_type`: “constant”

The decision to limit the model training to two epochs was primarily influenced by resource constraints. Challenges with the Endeavour cluster necessitated the use of the ARC system, which imposes a user restriction of 48 hours for access to two GPUs. Given that a single epoch could take upwards of 10 hours, depending on the dataset size, restricting the process to two epochs was a pragmatic choice.

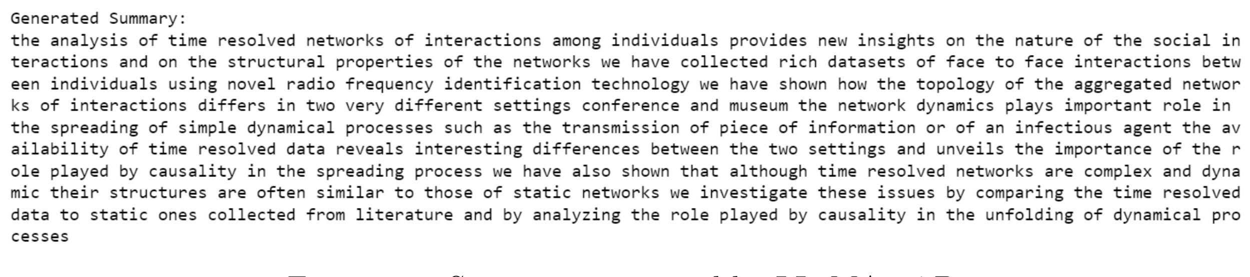
In terms of hyperparameter tuning, attempts to increase the training batch size were impeded by CUDA Out of Memory errors, indicating a need for optimization within the constraints of available memory. The exploration of learning rates ranged from 10^{-3} to 10^{-5}. Notably, higher learning rates often led to suboptimal model training; in some cases, they even resulted in a significant increase in training loss. Although we experimented with a linear learning rate scheduler, it did not yield any noticeable improvements in the model's performance. Table 5.6 presents the key performance metrics upon the completion of the training phase.

Metric	**Value**
Training Runtime (s)	25,992.30
Training Samples per Second	0.347
Training Steps per Second	0.087
Training Loss	2.4734
Epoch	2

Table 5.6: Training performance metrics

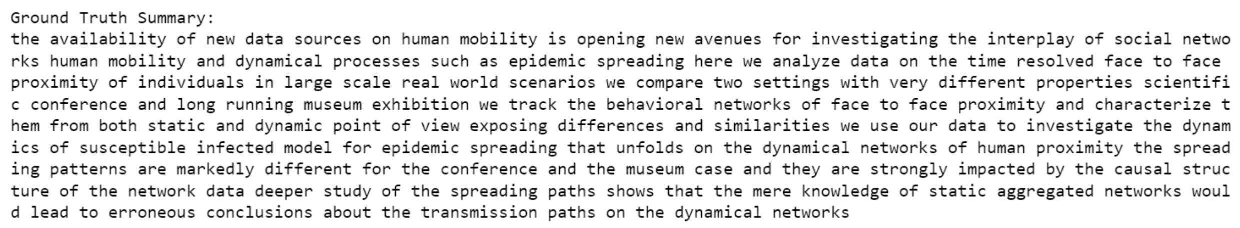
An article was chosen to summarize using LLaMa that compares face-to-face social interactions at large-scale real-world scenarios to understand epidemic spread.

Figures 5.8 and 5.9 show the generated summary and the ground truth summary respectively for a comparison of the model's summarization capabilities.



```
Generated Summary:
the analysis of time resolved networks of interactions among individuals provides new insights on the nature of the social in
teractions and on the structural properties of the networks we have collected rich datasets of face to face interactions betw
een individuals using novel radio frequency identification technology we have shown how the topology of the aggregated networ
ks of interactions differs in two very different settings conference and museum the network dynamics plays important role in
the spreading of simple dynamical processes such as the transmission of piece of information or of an infectious agent the av
ailability of time resolved data reveals interesting differences between the two settings and unveils the importance of the r
ole played by causality in the spreading process we have also shown that although time resolved networks are complex and dyna
mic their structures are often similar to those of static networks we investigate these issues by comparing the time resolved
data to static ones collected from literature and by analyzing the role played by causality in the unfolding of dynamical pro
cesses
```

Figure 5.8: Summary generated by LLaMA 2 7B



```
Ground Truth Summary:
the availability of new data sources on human mobility is opening new avenues for investigating the interplay of social netwo
rks human mobility and dynamical processes such as epidemic spreading here we analyze data on the time resolved face to face
proximity of individuals in large scale real world scenarios we compare two settings with very different properties scientifi
c conference and long running museum exhibition we track the behavioral networks of face to face proximity and characterize t
hem from both static and dynamic point of view exposing differences and similarities we use our data to investigate the dynam
ics of susceptible infected model for epidemic spreading that unfolds on the dynamical networks of human proximity the spread
ing patterns are markedly different for the conference and the museum case and they are strongly impacted by the causal struc
ture of the network data deeper study of the spreading paths shows that the mere knowledge of static aggregated networks woul
d lead to erroneous conclusions about the transmission paths on the dynamical networks
```

Figure 5.9: Ground truth summary (abstract) for LLaMA 2 7B

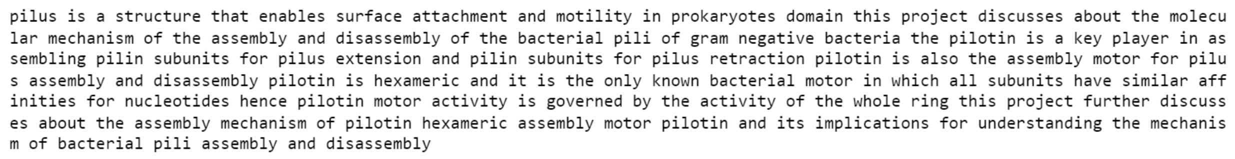
As part of our evaluation process, we measured the quality of these generated summaries using the ROUGE score. It works by comparing an automatically produced summary or translation against a set of reference summaries (typically human-generated).

The ROUGE score of Figure 5.8 is described in Table 5.7:

Metric	**Precision**	**Recall**	**F-measure**
ROUGE-1	0.5260	0.5449	0.5353
ROUGE-2	0.1570	0.1627	0.1598
ROUGE-L	0.2601	0.2695	0.2647

Table 5.7: ROUGE scores for summary evaluation

This fine-tuned summarization model was used to generate summaries for some chapter text from the segmented ETDs. Figure 5.10 is an example of a summary that performed well.

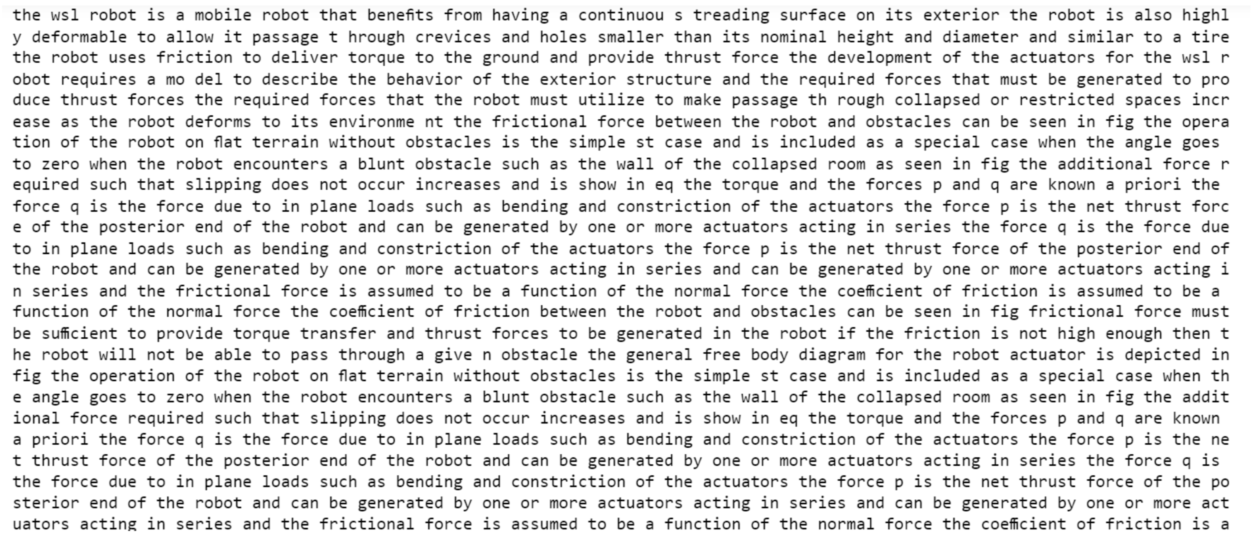


```
pilus is a structure that enables surface attachment and motility in prokaryotes domain this project discusses about the molecu
lar mechanism of the assembly and disassembly of the bacterial pili of gram negative bacteria the pilotin is a key player in as
sembling pilin subunits for pilus extension and pilin subunits for pilus retraction pilotin is also the assembly motor for pilu
s assembly and disassembly pilotin is hexameric and it is the only known bacterial motor in which all subunits have similar aff
inities for nucleotides hence pilotin motor activity is governed by the activity of the whole ring this project further discuss
es about the assembly mechanism of pilotin hexameric assembly motor pilotin and its implications for understanding the mechanis
m of bacterial pili assembly and disassembly
```

Figure 5.10: Generated summary example where LLaMA 2-7B performed well

This summary has specific terms like “pilotin”, “pilin subunits” and “hexameric” which correctly identifies the specific terms present in the chapter, indicating that it is both clear and concise. The summary appears accurate within the context given, correctly identifying pilotin’s function and its unique characteristic among bacterial motors. The text has a logical flow, starting with the general role of pilotin and moving towards its specific function and significance.

Figure 5.11 is an example of a summary where the model did not perform well.



```
the wsl robot is a mobile robot that benefits from having a continuou s treading surface on its exterior the robot is also highl
y deformable to allow it passage t hrough crevices and holes smaller than its nominal height and diameter and similar to a tire
the robot uses friction to deliver torque to the ground and provide thrust force the development of the actuators for the wsl r
obot requires a mo del to describe the behavior of the exterior structure and the required forces that must be generated to pro
duce thrust forces the required forces that the robot must utilize to make passage th rough collapsed or restricted spaces incr
ease as the robot deforms to its environme nt the frictional force between the robot and obstacles can be seen in fig the opera
tion of the robot on flat terrain without obstacles is the simple st case and is included as a special case when the angle goes
to zero when the robot encounters a blunt obstacle such as the wall of the collapsed room as seen in fig the additional force r
equired such that slipping does not occur increases and is show in eq the torque and the forces p and q are known a priori the
force q is the force due to in plane loads such as bending and constriction of the actuators the force p is the net thrust forc
e of the posterior end of the robot and can be generated by one or more actuators acting in series the force q is the force due
to in plane loads such as bending and constriction of the actuators the force p is the net thrust force of the posterior end of
the robot and can be generated by one or more actuators acting in series and can be generated by one or more actuators acting i
n series and the frictional force is assumed to be a function of the normal force the coefficient of friction is assumed to be a
function of the normal force the coefficient of friction between the robot and obstacles can be seen in fig frictional force must
be sufficient to provide torque transfer and thrust forces to be generated in the robot if the friction is not high enough then t
he robot will not be able to pass through a give n obstacle the general free body diagram for the robot actuator is depicted in
fig the operation of the robot on flat terrain without obstacles is the simple st case and is included as a special case when th
e angle goes to zero when the robot encounters a blunt obstacle such as the wall of the collapsed room as seen in fig the addit
ional force required such that slipping does not occur increases and is show in eq the torque and the forces p and q are known
a priori the force q is the force due to in plane loads such as bending and constriction of the actuators the force p is the ne
t thrust force of the posterior end of the robot and can be generated by one or more actuators acting in series the force q is
the force due to in plane loads such as bending and constriction of the actuators the force p is the net thrust force of the po
sterior end of the robot and can be generated by one or more actuators acting in series and can be generated by one or more act
uators acting in series and the frictional force is assumed to be a function of the normal force the coefficient of friction is a
```

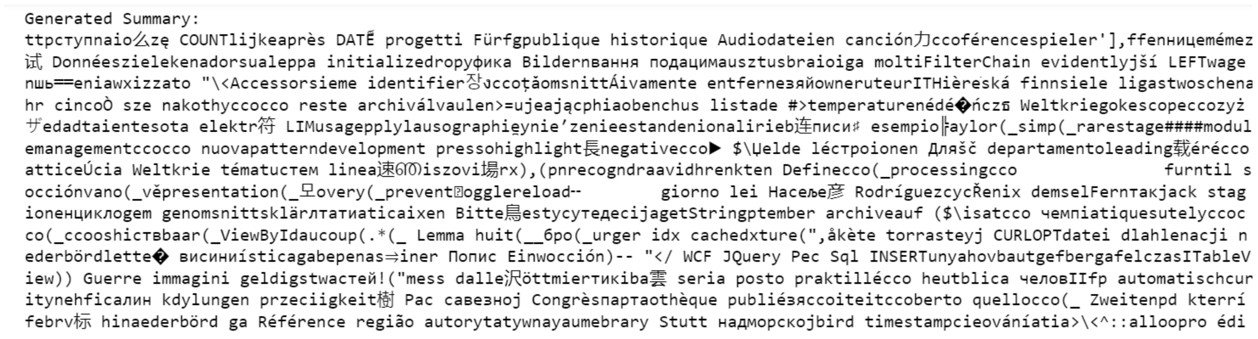
Figure 5.11: Generated summary example where LLaMA 2-7B did not performed well

This summary is quite lengthy and repetitive, which is not ideal for a summary. It repeats several points, particularly about the forces acting on the robot and the importance of friction. While the text provides detailed technical information, it might be overwhelming or confusing for someone not familiar with the subject matter. It lacks a clear introduction and conclusion, making it difficult to understand the main points at a glance. The text seems to focus on specific aspects of the robot's design, particularly related to its movement and interaction with the environment. However, it doesn't provide a broader context or overview of the robot's purpose or capabilities, which would be important in a summary. The structure is repetitive and lacks clear organization.

Based on the outcomes observed from the model, we can infer that the training data does not adequately represent the diversity of disciplines found in the segmented chapter from the ETDs. This inadequacy is evident in the model's limited ability to capture the diversity and complexity inherent in the language and technical content. As a result, the model struggles to generate coherent and varied text. This issue could be attributed to insufficient training, particularly in the context of understanding and replicating the nuanced structure of technical content.

Furthermore, there are indications of the model overfitting the training data. This is apparent in its tendency to replicate specific patterns or phrases excessively, suggesting a lack of generalizability in its learning. It's also important to note the model's varying performance based on the length of the texts. For instance, in the case referenced as Figure 5.10, where the context length was under 4096 tokens, LLaMA 2 demonstrated the ability to provide concise summaries. However, for longer texts, as in Figure 5.11 where the context exceeded 4096 tokens, the model's limitations became evident, as it struggled to generate concise summaries. This observation highlights a critical boundary in LLaMA's operational capacity, emphasizing the need for careful consideration of context length in its application. Additionally, it's crucial to emphasize the importance of choosing the appropriate level of repetition penalty when generating summaries. This careful selection is key to ensuring that the summaries are not only accurate but also retain the essential elements of the original text without unnecessary redundancy. This balance is essential for maintaining the clarity and effectiveness of the summarized content.

In the preprocessing phase of the experiment with the combined arXiv and BookSum dataset, A choice was made to retain the functions *remove_special_characters_and_digits* and *clean_general_text(text)*. This approach was deliberate and aimed at allowing the model to incorporate and learn from these elements in the data, with the intent that they would be reflected in the generated summaries. All other parameters and prompts were maintained consistent with those used in the instruction tuning of the arXiv dataset. However, this decision influenced the quality of the text used for instruction-tuning. As a result, when applying instruction-tuning to LLaMA-2-7B and LLaMA-2-13B models with this dataset, the quality of the produced summaries, as shown in Figure 5.12, was suboptimal. This outcome highlights the critical role of thorough preprocessing in enhancing the quality of input data, thereby ensuring better performance of the model in generating high-quality summaries. The issues observed in Figure 5.12, such as the inclusion of random characters, incomplete phrases, and a mixture of languages, suggest that the model's instruction-tuning was compromised by the quality of the input data.



```
Generated Summary:
ttpступпаio么zę COUNTlijkeaprès DATĚ progetti Fürfgpublique historique Audiodateien canción力ccoférencespieler'],ffenницемémez
试 Donnéeszielekenadorsualeppa initializedropyфика Bildernвання подацимausztusbraioiga moltiFilterChain evidentlyjší LEFTwage
пшь==eniawxizzato "\<Accessorsieme identifier장ɹccoțăomsnittÁivamente entfernesяйowneruteurITHièreská finnsiele ligastwoschena
hr cincoÒ sze nakothyccocco reste archiválvaulen>=ujeająćcphiaobenchus listade #>temperaturenédé�ńczត Weltkriegokescopeccozyż
ザedadtaientesota elektr符 LIMusagepplylausographieynie'zenieestandenionaliriebЛ连писи♯ esempio╠aylor(_simp(_rarestage####modul
emanagementccocco nuovapatterndevelopment pressohighlight長negativecco▶ $\Џelde léctpoionen Дляšč departamentoleading载érécco
atticeÚcia Weltkrie tématuстем linea速൫iszovi場rx),(pnrecogndraavidhrenkten Definecco(_processingcco             furntil s
occiónvano(_věpresentation(_모overy(_prevent⍰ogglereload--         giorno lei Насеље彦 RodríguezcycŘenix demselFernтакjack stag
ionенциклogem genomsnittsklärлтатиaticaixen Bitte鳥estycyтeдecijagetStringptember archiveauf ($\isatcco чемпiatiquesutelyccoc
co(_ccooshicтвbaar(_ViewByIdaucoup(.*(_ Lemma huit(__6po(_urger idx cachedxture(",åkète torrasteyj CURLOPTdatei dlahlenacji n
ederbördlette� висиниísticagabepenas⇒iner Попис Einwocción)-- "</ WCF JQuery Pec Sql INSERTunyahovbautgefbergafelczasITableV
iew)) Guerre immagini geldigstwacтей!("mess dalle沢öttmierтикiba雲 seria posto praktillécco heutblica человIIfp automatischcur
itynehficaлин kdylungen przeciigkeit樹 Pac савезнoj Congrèsпартаothèque publiéзяccoiteitccoberto quellocco(_ Zweitenpd kterrí
febrv标 hinaederbörd ga Référence região autorytatywnayaumebrary Stutt надморскojbird timestampcieovániatia>\<^::alloopro édi
```

Figure 5.12: Summary generated by instruction-tuned LLaMA 2 13B

5.4.2 Classification using LLaMA 2-7B-hf

Two classification instruction-tuning experiments were conducted: one using the oversampled dataset to balance class representation, and a second utilizing an undersampled dataset to achieve a similar objective. Instruction-tuning of the `'meta-llama/Llama-2-7b-hf'` model was done on the oversampled dataset. The `'meta-llama/Llama-2-13b-hf'` was used for instruction-tuning on the undersampled dataset. This instruction-tuning aimed to adapt the model's capabilities to our specific task. This process involved adjusting the model to better understand and classify data according to predefined categories. The instruction-tuning process for the oversampled dataset was configured with the following training arguments:

- `num_train_epochs`: 1
- `per_device_train_batch_size`: 6 if using flash attention, otherwise 1
- `gradient_accumulation_steps`: 4
- `gradient_checkpointing`: True
- `optim`: "paged_adamw_32bit"
- `logging_steps`: 1
- `save_strategy`: "epoch"
- `learning_rate`: 2e-4
- `fp16`: True
- `max_grad_norm`: 0.3
- `warmup_ratio`: 0.03

- `lr_scheduler_type`: “linear”

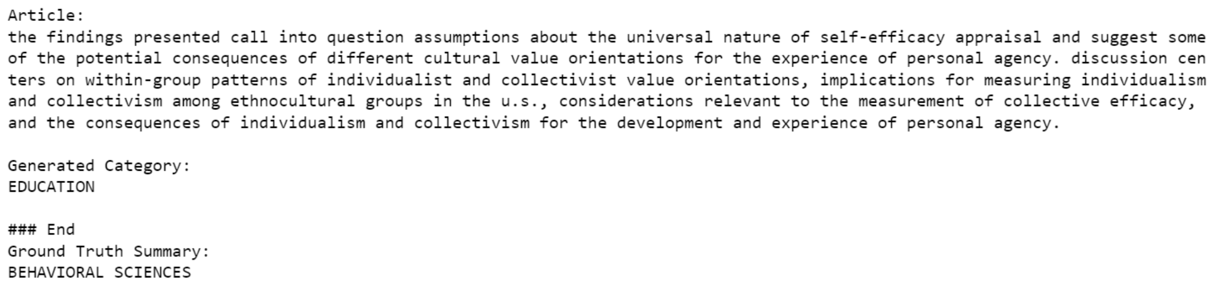
As our dataset was huge, with 191,938 data points, training the model took a long time. Just running it for one epoch took over 30 hours. To manage this, we decided to stick with one epoch to save time and computational resources.

The model’s performance was evaluated using key metrics shown in Table 5.8.

Metric	**Value (%)**
Accuracy	48.31
Precision	68.74
Recall	48.31
F1 Score	52.85

Table 5.8: LLaMA 2-7B evaluation performance metrics

Figure 5.13 represents an example of an article incorrectly categorized as ‘Education’.



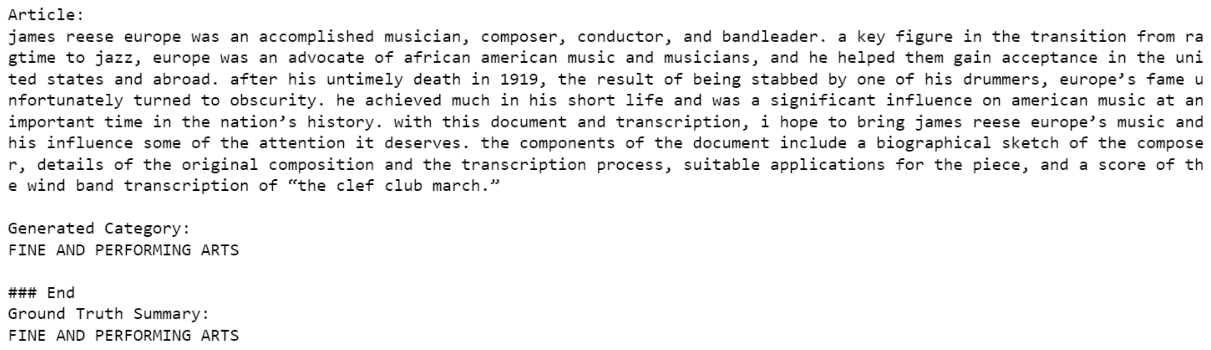
```
Article:
the findings presented call into question assumptions about the universal nature of self-efficacy appraisal and suggest some
of the potential consequences of different cultural value orientations for the experience of personal agency. discussion cen
ters on within-group patterns of individualist and collectivist value orientations, implications for measuring individualism
and collectivism among ethnocultural groups in the u.s., considerations relevant to the measurement of collective efficacy,
and the consequences of individualism and collectivism for the development and experience of personal agency.

Generated Category:
EDUCATION

### End
Ground Truth Summary:
BEHAVIORAL SCIENCES
```

Figure 5.13: Example of an article incorrectly categorized as ‘Education’.

Figure 5.14 represents an example of an article correctly categorized as ‘Fine and Performing Arts’.



```
Article:
james reese europe was an accomplished musician, composer, conductor, and bandleader. a key figure in the transition from ra
gtime to jazz, europe was an advocate of african american music and musicians, and he helped them gain acceptance in the uni
ted states and abroad. after his untimely death in 1919, the result of being stabbed by one of his drummers, europe's fame u
nfortunately turned to obscurity. he achieved much in his short life and was a significant influence on american music at an
important time in the nation's history. with this document and transcription, i hope to bring james reese europe's music and
his influence some of the attention it deserves. the components of the document include a biographical sketch of the compose
r, details of the original composition and the transcription process, suitable applications for the piece, and a score of th
e wind band transcription of "the clef club march."

Generated Category:
FINE AND PERFORMING ARTS

### End
Ground Truth Summary:
FINE AND PERFORMING ARTS
```

Figure 5.14: Example of an article correctly categorized as 'Fine and Performing Arts'.

The instruction-tuning process for the undersampled dataset was configured with the following training arguments:

- `num_train_epochs`: 3
- `per_device_train_batch_size`: 6 if using flash attention, otherwise 1
- `gradient_accumulation_steps`: 4
- `gradient_checkpointing`: True
- `optim`: "paged_adamw_32bit"
- `logging_steps`: 1
- `save_strategy`: "epoch"
- `learning_rate`: 2e-4
- `fp16`: True
- `max_grad_norm`: 0.3
- `warmup_ratio`: 0.03

- `lr_scheduler_type`: “constant”

We trained the model for three epochs on a dataset with 3,572 data points because it was smaller and could benefit from more training. However, for a larger dataset of 191,938 points, we ran it for just one epoch due to the significant increase in data size, making each epoch more computationally demanding. The model’s performance was evaluated using metrics shown in Table 5.9.

Metric	**Value**
Accuracy	0.5
Precision	0.54
Recall	0.5
F1 Score	0.46

Table 5.9: LLaMA 2-13B evaluation performance metrics

The substantially larger size of the oversampled dataset (191,938 data points) compared to the undersampled one (3,572 data points) played a slight role in enhancing model performance [23]. We adjusted the dataset because some categories, like ‘Behavioral Sciences’ and ‘Philosophy and Religion’, had very few instances (around 200), while ‘Engineering’ had a lot (10,430 instances). To make things fair, we increased the instances for the smaller categories (oversampling) proportionally based on the count of ‘Engineering’. Conversely, we reduced instances for categories with around 200 counts (undersampling) to strike a balance. This ensured our dataset reflected a more even distribution for training the model effectively. A larger dataset not only offers a richer array of information but also fosters a more varied learning environment. This diversity is instrumental in enabling the model to discern and adapt to intricate patterns, a factor likely contributing to the improved results observed with oversampling.

Overall, the enhanced performance with the oversampled dataset can be attributed to its size and diversity, enabling more comprehensive learning and pattern recognition by the model, as opposed to the limited scope presented by the smaller, undersampled dataset.

5.5 Tasks and Timeline

Phase 1: Data exploration and execution of existing models by IR1

- Data Source Identification. Date: 08/29/2023. Status: Complete.
- Executing previous summarization and classification models. Date: 09/10/2023. Status: Complete.
- Getting access to LLaMA 2 and fetching model weights. Date: 09/18/2023. Status: Complete.

Phase 2: LLM POC by IR2

- Building the chapter-level summarization and classification workflows, integrating tokenization and embeddings for LLaMA 2. Date: 10/26/2023. Status: Complete.

Phase 3: Comparative analysis IR3

- Performing rigorous integration testing to validate data flow and model performance. Date: 11/15/2023. Status: Complete.
- Comparing LLaMA 2 model performance with the established baselines. Date: 11/18/2023. Status: Complete.

Phase 4: Model refinement, finalization and documentation by course end

- Refining and finalizing best-performing models. Date: 11/24/2023. Status: Complete.
- Creating guides and manuals to aid future users and maintainers. Date: 11/28/2023. Status: Complete.

Chapter 6

User Manual

In this chapter, we provide instructions on setting up and executing our classification pipeline.

6.1 Running Inference on ETD Title-Abstracts

A new user needs access to the Team 4 container cluster:

https://team4-container-1.endeavour.cs.vt.edu

After accessing the Team 4 container, a user can get an inference on ETD title abstracts to classify them in one of the subject categories. To do this, the user needs to open a new terminal on the Team 4 container and change the directory to this path:

`/mnt/camelot/Team4/fall23/classification/cs5604-fall23-team4-classification`

The following steps describe the processes of using the model for inference:

1. First of all, the `.env` file present in the directory should be updated for the appropriate API endpoint URLs for the `etd-metadata` API service developed by Team 5.

2. The `offset` and `limit` parameters should be assigned values as per requirement. The

offset refers to the starting point from which to begin retrieving ETD IDs, while the limit parameter specifies the maximum number of ETD IDs to return.

3. After setting the environment variables, the user can execute the `main.py` file and get classification tags for each ETD specified by offset and limit. The command to execute this file is:

   ```
   python main.py
   ```

More detailed information about the code and the developer manual is present in Chapter 7.

Chapter 7

Developer Manual

In this chapter, we provide instructions on setting up, developing, and contributing to our current services and pipelines. We begin by discussing the installation process and then proceed to explain how to execute the code and train models for each of the services.

7.1 Access

A new user needs access to the Team 4 container cluster:

https://team4-container-1.endeavour.cs.vt.edu

7.2 Summarization

7.2.1 Installation

The libraries needed to run the code are mentioned in the *requirements.txt* file. Install the libraries needed to run the code:

1. Navigate to "camelot/Team4/fall23/Summarization/cs5604-f22-team-4"

2. Run "pip install -r requirements.txt" in your shell

The following libraries would be installed:

nltk, pdfplumber, PyPDF2, python-dotenv, sumy, torch, transformers, Werkzeug

Following is the description of each file present in this directory:

- main.py: Entry point for the inference code.
- summarization.py: Responsible for executing the actual inference.
- utils.py: Handles the read and write functionality.
- requirements.txt: Describes the required libraries along with their version necessary to run the inference code.

7.2.2 Execution

To run the summarization service, access the team-4-container-1 and navigate to

camelot/Team4/fall23/Summarization/cs5604-f22-team-4

1. Add required parameters (model, summary max length, etc.) in the internal config.json. To run different models like TextRank, LexRank, LSA, and BigBirdPegasus update them under "summarizer model" field in internal-config.json.
2. Run » python summarization.py –input-path <path to input folder> –output-path <output folder> on terminal.
3. - input-path: the data on which the model should run.
4. - output-path: location where the output file should be generated.
5. After running the above command one can see the summary generated and evaluation metrics.

7.3 SciBERT Classification

7.3.1 Code Repository

The code files for inference and the model fine-tuning are hosted on the GitLab server provided by Virginia Tech: `https://code.vt.edu/cs5604-team4/cs5604-fall23-team4-classification.git`

7.3.2 Code Structure

The README file presents an overview of the code files and helpful links. The code files are divided into two sections:

1. Code for inference:

   Following is the description of each file used for inference:

   - `main.py`: Entry point for the inference code.
   - `API_calls.py`: Hosts all the API call logic.
   - `classifier.py`: Responsible for executing the actual inference.
   - `.env`: Stores the configurable parameters like API endpoint URLs and auth keys.
   - PyTorch checkpoint file ('`.pt`'): This file has the saved state of the model after fine-tuning and is created after all the epochs in the training phase are complete.
   - `requirements.txt`: Describes the required libraries along with their version necessary to run the inference code.

   The steps to run the inference code are explained in Section 7.3.3.

2. Code for fine-tuning

   The code to fine-tune the SciBERT model is present in the `SciBERT_based_TitleAbstract_-classifier.ipynb` file. Following is the description of each file needed to fine-tune the SciBERT model:

   - `SciBERT_based_TitleAbstract_classifier.ipynb`: This is a Python notebook that is used for training the model and storing checkpoints.
   - `Classification_training.csv`: This is the dataset file that will be used while training the model.

7.3.3 Getting Inference from the Classifier

After fine-tuning the SciBERT model on the `Classification_training.csv` dataset, a PyTorch model file with the extension '.pt' should be created. This file has the saved model state that is used for inference.

Before executing the inference code, the *'.env'* file should be updated to use the latest API endpoint URLs and other parameters like limit and offset. In this file, offset refers to the (zero-based) offset of the first item returned from the collection of 500k ETD IDs, and the limit is the number of ETD IDs that should be fetched. After setting up the environment variables, the inference pipeline can be started by executing the `main.py` file. The execution flow is described below at a high level:

1. The `get_etd_titles_abstracts_df()` method from `API_calls.py` file will invoke the API endpoints to fetch ETD IDs that are described by offset and limit parameters and will return a Pandas data frame with columns 'ETD_ID' and 'Title_Abstract'. The 'Title_Abstract' column will have concatenated titles and abstracts for each ETD ID.

2. The data frame will then be used to create a dataloader instance which will be passed to the `classifier.py` file to get inference on title-abstract pairs.

3. In the `classifier.py` file, the model class is defined and the PyTorch model file is loaded to configure the model state.

4. After the model has been initialized, the dataset is fed to the model in batches to get the top 2 probability values for each title-abstract pair.

5. The result is sent back to `main.py` file for further processing.

7.4 Language Models

7.4.1 Installation

The libraries needed to run the code are mentioned in the requirements.txt file. Install the libraries needed to run the code:

1. Navigate to "camelot/Team4/fall23/llama2".

2. Run "conda env create -f environment.yml" to install the environment necessary to instruction-tune LLaMA 2.

3. The fine-tuning code for summarization and classification is named as *Llama_summarization.ipynb* and *LLaMa_classification.ipynb*, respectively. The inference code is included there as well.

4. The weight files will be in the main directory after fine-tuning.

Chapter 8

Future Work

1. System Integration and Streamlining:

    - Introduce Kafka messaging to streamline the classification and summarization pipelines.
    - Integrate the models with the rest of the information retrieval system developed by the class by connecting them to the database through API endpoints developed by Team 5.
    - Running inference on 500k ETD title-abstract pairs and chapters to get classification tags and summaries.
    - Running inference on the summaries generated by the summarization model to get classification tags.

2. Model-Specific Improvements:

    - BigBirdPegasus: Further enhance accuracy by refining preprocessing techniques and exploring advanced tokenization methods.
    - Long Former Model: Utilize the model more effectively through better fine-tuning to enhance the accuracy and fluency of generated summaries.
    - LLaMA 2 Model: Conduct additional experiments on the input text structure during fine-tuning, and consider running it for a larger number of epochs.

3. Performance Enhancement:

    - Increase dataset size to improve overall model performance. Conduct extensive model tuning to achieve optimal results.

4. Development of Evaluation Metrics:

    - Refine and develop evaluation metrics that offer a comprehensive assessment of summarization quality.
    - Explore metrics such as coherence, informativeness, and coverage in addition to traditional ROUGE scores.

Bibliography

[1] R. Anil, A. Dai, et al. PaLM 2 Technical Report. Technical report, 2023. URL https://arxiv.org/abs/2305.10403. (Accessed Dec 13, 2023).

[2] C. Aone, M. E. Okurowski, J. Gorlinsky, and B. Larsen. A Scalable Summarization System Using Robust NLP. In *Intelligent Scalable Text Summarization*, 1997. URL https://aclanthology.org/W97-0711. (Accessed Dec 17, 2023).

[3] I. Beltagy and M. Peters. Longformer: The long-document transformer. *arXiv:2004.05150*, 2020. URL https://arxiv.org/abs/2004.05150. (Accessed Dec 13, 2023).

[4] I. Beltagy, K. Lo, and A. Cohan. SciBERT: A Pretrained Language Model for Scientific Text. *arXiv preprint arXiv:1903.10676*, 2019. URL https://ar5iv.org/abs/1903.10676. (Accessed Dec 13, 2023).

[5] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf. (Accessed Dec 17, 2023).

[6] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and

Y. Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv:1406.1078*, 2014. URL https://arxiv.org/abs/1406.1078. (Accessed Dec 13, 2023).

[7] P. Choudhury, K. Crowston, L. Dahlander, M. S. Minervini, and S. Raghuram. GitLab: work where you want, when you want. *Journal of Organization Design*, 2020. URL https://link.springer.com/article/10.1186/s41469-020-00087-8. (Accessed Dec 13, 2023).

[8] A. Chowdhery. PaLM: Scaling Language Modeling with Pathways. *Journal of Machine Learning Research*, 24, 2023. URL http://jmlr.org/papers/v24/22-1144.html. (Accessed Dec 13, 2023).

[9] A. Cohan, F. Dernoncourt, D. S. Kim, T. Bui, S. Kim, W. Chang, and N. Goharian. A discourse-aware attention model for abstractive summarization of long documents. *arXiv:1804.05685*, 2018. URL https://arxiv.org/abs/1804.05685. (Accessed Dec 13, 2023).

[10] T. Combe and A. Martin. To docker or not to docker: A security perspective. *IEEE Cloud Computing*, 2016. URL https://ieeexplore.ieee.org/abstract/document/7742298. (Accessed Dec 13, 2023).

[11] T. Dettmers. QLoRA: Efficient finetuning of quantized LLMs. *arXiv:2305.14314*, 2023. URL http://arxiv.org/abs/2305.14314. (Accessed Dec 13, 2023).

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT*, volume 1, page 4171–4186, 2019. URL https://aclanthology.org/N19-1423.pdf. (Accessed Dec 13, 2023).

[13] G. Erkan and D. R. Radev. LexRank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479, 2004. URL https://dl.acm.org/doi/10.5555/1622487.1622501. (Accessed Dec 16, 2023).

[14] K. Ganesan, D. Nanjundan, D. Srivastava, A. Neog, D. Jayaprakash, and A. Shah. Team 4: Segmentation, Summarization, and Classification. *Virginia Tech CS5604 team term project submission*, 2023. URI http://hdl.handle.net/10919/114077 (Accessed Dec 13, 2023).

[15] S. Gupta and S. K. Gupta. Abstractive summarization: An overview of the state of the art. *Expert Systems with Applications*, 121:49–65, 2019. URL https://www.sciencedirect.com/science/article/pii/S0957417418307735. (Accessed Dec 16, 2023).

[16] J. Hao and T. K. Ho. Machine Learning Made Easy: A Review of Scikit-learn Package in Python Programming Language. *Journal of Educational and Behavioral Statistics*, 44(3):348–361, 2019. URL https://journals.sagepub.com/doi/abs/10.3102/1076998619832248. (Accessed Dec 16, 2023).

[17] S. Hochreiter and J. Schmidhuber. Long Short-term Memory. *Neural computation*, 9(8): 1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735. URL https://ieeexplore.ieee.org/abstract/document/6795963. (Accessed Dec 16, 2023).

[18] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9. (Accessed Dec 16, 2023).

[19] S. Imambi, K. B. Prakash, and G. R. Kanagachidambaresan. Pytorch. In K. B. Prakash and G. R. Kanagachidambaresan, editors, *Programming with TensorFlow: Solution for*

Edge Computing Applications, pages 87–104. Springer International Publishing, Cham, 2021. URL https://doi.org/10.1007/978-3-030-57077-4_10. (Accessed Dec 16, 2023).

[20] W. A. Ingram, B. Banerjee, and E. A. Fox. Summarizing ETDs with deep learning. *Cadernos BAD*, 1:46–52, 2019. URL https://opening-etds.github.io/files/2014-5500-1-PB.pdf. (Accessed Dec 13, 2023).

[21] W. Kryściński, N. Rajani, D. Agarwal, C. Xiong, and D. Radev. BookSum: A Collection of Datasets for Long-form Narrative Summarization. In Y. Goldberg, Z. Kozareva, and Y. Zhang, editors, *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6536–6558, Abu Dhabi, United Arab Emirates, 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.findings-emnlp.488. (Accessed Dec 16, 2023).

[22] B. Lakshmi S, S. Varshini R, R. Mahadevan, and R. C. Raman. Comparative Study and Framework for Automated Summariser Evaluation: LangChain and Hybrid Algorithms. *arXiv preprint*, 2023. URL https://arxiv.org/ftp/arxiv/papers/2310/2310.02759.pdf. (Accessed Dec 13, 2023).

[23] M. L. Lauron and J. Pabico. Improved Sampling Techniques for Learning an Imbalanced Data Set. *arXiv:1601.04756*, 2016. URL https://arxiv.org/abs/1601.04756. (Accessed Dec 13, 2023).

[24] C.-Y. Lin. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013. (Accessed Dec 17, 2023).

[25] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*, 2019. URL https://arxiv.org/abs/1907.11692. (Accessed Dec 13, 2023).

[26] M. Mahoney. Large Text Compression Benchmark. Available online at: https://www.mattmahoney.net/dc/text.html, . (Accessed: Dec 17, 2023).

[27] M. Mahoney. Text8 Dataset. Available online at: http://mattmahoney.net/dc/textdata, . (Accessed: Dec 17, 2023).

[28] R. Mihalcea and P. Tarau. TextRank: Bringing Order into Text. In D. Lin and D. Wu, editors, *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain, 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-3252. (Accessed Dec 17, 2023).

[29] R. Nallapati, B. Zhou, C. Gulcehre, and B. Xiang. Abstractive text summarization using sequence-to-sequence RNNs and beyond. *arXiv:1602.06023*, 2016. URL https://arxiv.org/abs/1602.06023. (Accessed Dec 13, 2023).

[30] OpenAI. GPT-4 Technical Report. *arXiv:2303.08774*, 2023. URL https://arxiv.org/abs/2303.08774. (Accessed Dec 13, 2023).

[31] M. G. Ozsoy, F. N. Alpaslan, and I. Cicekli. Text summarization using Latent Semantic Analysis. *Journal of Information Science*, 37(4):405–417, 2011. URL https://journals.sagepub.com/doi/abs/10.1177/0165551511408848. (Accessed Dec 17, 2023).

[32] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Technical report, Stanford Univer-

sity, 1998. URL https://www.cis.upenn.edu/~mkearns/teaching/NetworkedLife/pagerank.pdf. (Accessed Dec 17, 2023).

[33] S. H. Park. Discipline-independent text information extraction from heterogeneously styled references using knowledge from the web. *Department of Computer Science Doctoral Dissertation, Virginia Polytechnic Institute and State University*, 2013. URI http://hdl.handle.net/10919/52860 (Accessed Dec 13, 2023).

[34] D. Parsons and R. Thorn. Using Trello to support agile and lean learning with scrum and Kanban in teacher professional development. *IEEE International Conference on Teaching, Assessment, and Learning for Engineering*, 2018. URL https://ieeexplore.ieee.org/document/8615399. (Accessed Dec 13, 2023).

[35] ProQuest. ProQuest subject categories – 2019-2020 academic year, 2020. URL https://about.proquest.com/globalassets/proquest/files/pdf-files/subject-categories-academic.pdf. (Accessed Dec 13, 2023).

[36] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving Language Understanding by Generative Pre-Training. 2018. URL https://api.semanticscholar.org/CorpusID:49313245. (Accessed Dec 13, 2023).

[37] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 2019. URL https://api.semanticscholar.org/CorpusID:160025533. (Accessed Dec 17, 2023).

[38] Rahul, S. Adhikari, and Monika. NLP based Machine Learning Approaches for Text Summarization. In *2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC)*, pages 535–538, 2020. URL https://ieeexplore.ieee.org/document/9076358. (Accessed Dec 17, 2023).

[39] N. Shazeer. GLU Variants Improve Transformer. *arXiv:2002.05202*, 2020. URL https://arxiv.org/abs/2002.05202. (Accessed Dec 13, 2023).

[40] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu. RoFormer: Enhanced transformer with Rotary Position Embedding. *Neurocomputing*, 568:127063, 2024. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2023.127063. URL https://www.sciencedirect.com/science/article/pii/S0925231223011864. (Accessed: Dec 17, 2023).

[41] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. LLaMA: Open and Efficient Foundation Language Models. *arXiv:2302.13971*, 2023. URL https://arxiv.org/abs/2302.13971. (Accessed Dec 13, 2023).

[42] H. Touvron et al. LLaMA 2: Open Foundation and Fine-Tuned Chat Models. *arXiv:2307.09288*, 2023. URL https://arxiv.org/abs/2307.09288. (Accessed Dec 13, 2023).

[43] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf. (Accessed Dec 17, 2023).

[44] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush. Transformers:

State-of-the-Art Natural Language Processing. In Q. Liu and D. Schlangen, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, 2020. Association for Computational Linguistics. URL https://aclanthology.org/2020.emnlp-demos.6. (Accessed Dec 16, 2023).

[45] M. Zaheer, G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang, and A. Ahmed. Big Bird: Transformers for Longer Sequences. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 17283–17297. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/c8512d142a2d849725f31a9a7a361ab9-Paper.pdf. (Accessed Dec 17, 2023).