KTH ROYAL INSTITUTE
OF TECHNOLOGY

Degree Project in Computer Engineering

First cycle, 15 credits

Character Navigator

Automated Summarization of Characters in E-Books

DANIEL DAHLBERG
AXEL MÅNSON LOKRANTZ

Stockholm, Sweden, 2024

Character Navigator

Automated Summarization of Characters in E-Books

DANIEL DAHLBERG

AXEL MÅNSON LOKRANTZ

Degree Programme in Computer Engineering
Date: May 31, 2024

Supervisors: Niharika Gauraha, Nils Erlansson, Niklas Arbin
Examiner: Fredrik Kilander

Host company: BookBeat
Swedish title: Character Navigator
Swedish subtitle: Automatiserad summering av karaktärer i E-böcker

© 2024 Daniel Dahlberg and Axel Månson Lokrantz

Abstract

The advent of E-books has revolutionized book consumption but also introduced challenges. E-books offer digital benefits but lack the kinesthetic feedback of printed books, affecting the reading experience. This thesis aims to address these issues by developing a new digital feature for E-books: automatic summarization of characters based on reading progress to avoid spoilers.

Existing research has explored book-length summarization using state-of-the-art machine learning models, but studies on character summarization are limited and outdated. This thesis explores the use of Large Language Models (LLMs) for summarization of fictional characters and develops an interface to present these summaries, the latter being a previously unexplored area.

Three services are developed: a character summarization service, a server, and a client. The character summarization service identifies characters using the BERT-Large-NER model and summarizes them with GPT-4 using hierarchical merging. Summaries are stored as CSV files, accessed via the server, and displayed in the client through a panel activated by clicking a character's name. Only summaries up to the clicked character's location are shown to prevent spoilers.

The evaluation focuses on the machine learning service's ability to generate well-crafted summaries, assessed by content and format adherence. Content quality is measured using reference-based BERTScore, which calculates semantic similarity against a gold-standard summary. Format adherence is evaluated using a custom framework counting narratological components and their correct order. We found that while content adherence met the set threshold, format adherence results were unsatisfactory.

Keywords

E-books, Character Summaries, Machine Learning, Large Language Models, Named Entity Recognition, ML, LLM, NER

Sammanfattning

E-böcker har revolutionerat bokkonsumtionen men också introducerat utmaningar. E-böcker erbjuder digitala fördelar men saknar den kinestetiska återkopplingen som återfinns i tryckta böcker, vilket påverkar läsupplevelsen. Denna uppsats syftar till att lösa dessa problem genom att utveckla en ny digital funktion för E-böcker: automatisk sammanfattning av karaktärer baserat på användarens position i boken för att undvika spoilers.

Befintlig forskning har utforskat sammanfattning av böcker med moderna maskininlärningsmodeller, men forskning om sammanfattning av karaktärer är begränsade och ej aktuella. Denna uppsats utforskar användningen av stora språkmodeller för att sammanfatta skönlitterära karaktärer och utvecklar ett gränssnitt för att presentera dessa sammanfattningar, varav det senare är ett tidigare outforskat område.

Arbetet innefattar tre tjänster: en karaktärssammanfattningstjänst, en server och en klient. Karaktärssammanfattningstjänsten identifierar karaktärer med hjälp av modellen BERT-Large-NER och sammanfattar dem med GPT-4 genom hierarkisk sammanfogning. Sammanfattningarna lagras som CSV-filer, åtkomliga via servern, och visas i klienten genom en panel som aktiveras genom att klicka på en karaktärs namn. Endast sammanfattningar fram till den markerade karaktärens plats visas för att undvika spoilers.

Utvärderingen fokuserar på maskininlärningstjänstens förmåga att generera välskrivna sammanfattningar, bedömda efter innehålls- och formatöverensstämmelse. Innehållskvaliteten mättes med referensbaserad BERTScore, som beräknar semantisk likhet gentemot en guldstandardssammanfattning. Formatöverensstämmelse utvärderas med ett av författarna tillverkat ramverk som räknar narratologiska komponenter och deras korrekta ordning. Vi fann att innehållsöverensstämmelsen nådde den bestämda tröskeln, men resultaten för formatöverensstämmelse var otillfredsställande.

Nyckelord

E-böcker, Karaktärssammanfattningar, Maskininlärning, Stora språkmodeller, Namngiven entitetsigenkänning, ML, LLM, NER

Acknowledgments

We would like to express our deepest gratitude to our advisor, Niklas Arbin, for his support, belief in our ideas, and invaluable insights throughout the project. Words cannot express our gratitude for the opportunity he provided us to write this thesis at BookBeat.

We would also like to thank our supervisor, Niharika Gauraha, for her continuous feedback on the thesis that she gave us through the many seminars that were held throughout the project. Our examiner, Fredrik Kilander, also provided us with invaluable advice at the outset of the project in regards to the project's scope, for which we are deeply grateful.

Additionally, we extend our appreciation to Nils Erlansson for patiently listening to our ideas and providing crucial feedback and insights into the machine learning tasks, which helped our project exceed our expectations.

We also extend special thanks to Göran Sandström for assisting us in developing an effective plan for evaluating the project. Lastly, our sincere thanks go to Runa Torbacke for taking such good care of us during our time at BookBeat.

Stockholm, May 2024
Daniel Dahlberg Axel Månson Lokrantz

Contents

List of Figures

List of Tables

Listings

List of acronyms and abbreviations

CFI	Canonical Fragment Identifier
ePub	Electronic Publication
KTH	KTH Royal Institute of Technology
LLM	Large Language Model
NER	Named Entity Recognition
ROUGE	Recall-Oriented Understudy for Gisting Evaluation

Chapter 1

Introduction

The work conducted for this thesis had two primary objectives: First, to explore the application of machine learning in generating character summaries based on how far the reader has progressed in an E-book; Second, to deliver these summaries via clickable character names within the E-book interface. The accomplishment of these objectives is anticipated to assist readers in recalling characters without risking exposure to spoilers.

1.1 Background

Digital books, i.e. E-books and audio books, have increasingly gained traction in Sweden in recent years [1]. While these digital formats present novel opportunities for readers, they also introduce challenges. On the one hand, the mobile applications in which digital books are read typically offer features like in-app dictionaries for looking up unfamiliar expressions when highlighted by the reader, as well as note-taking capabilities to assist readers in recalling plot details. On the other hand, studies [1, 2, 3] suggest that digital books, both in E-book and audio book formats, impedes readers' ability to construct cognitive maps of stories.

Cognitive maps here refers to the spatial representation of literary elements that constitute the story. Mangen et. al [2] has attempted to define such literary elements as: (i) *characters*: their physical appearance, personality traits and relations to other characters; (ii) *geographical setting*: the locations of the story; (iii) *key locations*: locations significant to the plot; (iv) *objects*: objects significant to the plot; (v) *time and temporality*: time lapse between events and their duration, as well as chronology.

The reason for this impairment of constructing cognitive maps of stories is

explained in some studies [2, 3] as the absence of kinesthetic feedback inherent in the physical interaction with printed books. In other words, printed books provide both visual and physical points of reference —visual in the sense of viewability of page numbers; physical in the sense of the thickness and weight of the pages to the left and right of the current page, the flipping of pages, and so forth —that can be utilized to navigate to the literary elements mentioned previously. In contrast, digital book readers only have visual information to rely on.

1.2 Problem

1.2.1 Original Problem and Definition

In section 1.1, the opportunities and challenges of the digital book format are discussed, where the opportunities are identified as digital affordances that help readers in digesting the book, and the challenges as the impairment of the reader's ability to construct cognitive maps of stories. This work assumed that these challenges can be overcome by taking advantage of the opportunities, i.e. by providing digital affordances for retrieving information that readers struggle to track due to the lack of kinesthetic feedback. Specifically, we assumed that providing the readers with machine learning generated summaries of the literary elements outlined in the section 1.1 is an adequate method of addressing the challenges. However, given the complexity of the task and the timeframe of the project, the project was confined to only handling summaries of one of the literary elements, namely characters, for one specific digital book format in one language: E-books written in English.

Therefore, the problem statement was defined as follows: how can machine learning summarize characters within English E-books in a spoiler-free manner?

1.2.2 Scientific and Engineering Issues

The work delved into various computer engineering-related fields to draw inspiration on how to solve subissues of the problem that is delineated in section 1.2.1. The issues were the following:

1. **Prompt engineering related issues**: As explained in section 3.2.2, the work utilized Large Language Models (LLMs) to generate the character summaries. To clarify the issues associated with LLMs, some

terminology must be introduced. This terminology is covered in detail in section 2.1 and briefly explained in the following paragraphs.

LLMs generate their output in response to *prompts*, which are user-provided instructions formulated in natural language. A prompt not only contains instructions but may also incorporate *context* that the LLM considers when producing its response. In this work, every prompt comprised both elements: the instructions direct the LLM on how to craft the character summaries, while the context is provided by the source text from the book that the summary is based on. Nevertheless, these prompts are constrained by certain limitations inherent to the LLM, each bringing its own set of challenges:

(a) *The prompt must not exceed the LLM's context window.*

Context window refers to limitations on the maximum size of the prompt that can be processed by the LLM. Typically, the instructional component of a prompt is small in size; however, the ideal context often exceeds the allowed context window. In instances where the preferred context is too voluminous to fit within this window, it must be segmented into smaller "chunks". We refer to this segmentation process as *chunkification* for ease of reference in later chapters.

To appreciate the challenges posed by this limitation, it is important to consider two additional criteria that, while not mandatory, enhance the probability of producing high-quality summaries: (i) Having a large context increases the chances of the LLM extracting relevant information for the summary; (ii) The selected context should ideally conclude at a natural break point within the book. If the context is truncated without regard for these natural break points, it may impair the LLM's ability to link crucial details with the relevant character, especially if the character is mentioned in one chunk and the significant information in another. To put it differently, for the purposes of this work, the context should avoid ending mid-sentence and should preferably encompass complete narrative units such as entire sentences, paragraphs, or chapters. In this thesis, these narrative units are referred to by their linguistically proper term *discourse units*. When a chunk is only constituted by a certain discourse unit, it is referred to as a *cohesive chunk*, whereas a chunk that has does not is an *incohesive chunk*.

The challenge lies in the conflict between conditions (i) and (ii): While condition (i) suggests that the context should be as extensive as possible within the context window, condition (ii) recommends that the context comprise whole discourse units. Given that the length of discourse units can vary significantly, it becomes challenging to predetermine the optimal number of such units that will maximize the use of the context window.

(b) *Output consistency requires explicit instructions*.

The output from an LLM can vary significantly from one prompt to another unless the prompt precisely specifies the desired format of the output. To ensure uniformity in generating character summaries, it is essential to develop standardized templates for the prompts. While there is an abundance of guides [4, 5, 6] and research [7] on how to make the LLM's response to the prompt behave as intended —which are partially examined in sections 2.1, 7.2 and 7.3 —targeted case studies that investigate the development of prompt templates specifically for character summaries seem to be limited.

2. **Named Entity Recognition (NER)**: This work employed Named Entity Recognition (NER) models —models that are trained to identify entities within a text —to detect characters in E-books. However, even the most advanced NER models can struggle with identifying entities.

3. **Cost issues**: Generating text from extensive documents is a time-intensive process, but also a financially demanding one, as LLMs necessitate substantial computational resources.

4. **Performance of book-length summarization**: Although there already exist well-performing algorithms for summarizing book-length documents [8], there appears to be no algorithms of equal performance for summarizing characters in books specifically, as detailed in section 2.3.

5. **User interface related issues**: To the best of our knowledge, there are no research on how character summaries can be presented through the same interface as the E-book reader, let alone character summaries that are incrementally updated as the story progresses.

1.3 Purpose

The purpose of the work can be viewed from two perspectives: from an engineering standpoint, and from the standpoint of the company for which the work was conducted.

1.3.1 Purpose from an Engineering Standpoint

The purpose seen from an engineering standpoint was to contribute knowledge on how machine learning can be applied to E-books to generate character summaries.

1.3.2 Purpose from the Company's Standpoint

Although the challenges highlighted in section 1.2.1 are not limited to a specific demographic, they hold particular relevance for Swedish junior high school students for two reasons.

Firstly, the major consumers of audio books within Sweden are individuals born in the 2000s, a demographic to which a majority of junior high school students belong, and the medium's popularity within that demographic continues to rise [1]. Secondly, the PISA 2022 assessment [9] indicates a concerning decline in reading literacy among Swedish junior high school students. These challenges underscore the significance of finding solutions, and this is where BookBeat, a prominent digital book streaming service located in Sweden, plays a pivotal role.

BookBeat caters not only to the broader audience of digital fiction consumers but also collaborates with Swedish schools, providing students with digital formats of books assigned for class readings. Addressing these challenges aligns with BookBeat's mission to enhance the reading experience of their customers, particularly in the educational context.

Therefore, the purpose seen from the company's standpoint is to provide a better user experience to their customers.

1.4 Goals

In this thesis, the terms *objectives* and *goals* differ. *Objectives* refer to the tasks listed at the beginning of chapter 1, representing the objectives from the company's standpoint, specifically the functionalities we committed to

implement for BookBeat. These functionalities were not required to be evaluated by the company.

In contrast, *goals* refer to the tasks outlined in this section. These were the functionalities we aimed to both implement and evaluate to meet KTH Royal Institute of Technology (KTH)'s requirements for assessing this work. In other words, while the implementation tasks overlap, the goals are a subset of the objectives that were specifically evaluated to contribute knowledge to the scientific community.

The goals are derived from the issues presented in section 1.2.2. Although solutions to all the issues were implemented as part of the objectives, only points 1-b and 4 were subject to evaluation.

- **Solution to 1-b**: Prompt templates tailored to produce consistent character summaries were written. The character summaries' adherence to the summary structure was qualitatively analyzed to determine if the templates worked as intended.

- **Solution to 4**: An existing algorithm for book-length summarization, known as hierarchical merging, was adapted and adjusted to be applicable to character summaries. Moreover, the generated character summaries must meet a quality threshold measured by BERTScore, an evaluation metric detailed in section 2.5.1.2 and justified in section 2.5.3. In this work, sufficient quality was defined as achieving a mean BERTScore F1 score of at least 0.6, with 95% confidence that the BERTScore F1 score will not fall below 0.5. The justification for these thresholds are provided in section 2.5.4

1.5 Research Methodology

This section outlines the foundational assumptions that guided us in our methodological choices for this thesis.

The method selection process was informed by our prior experience with machine learning. While familiar with the basic principles to analyze and employ machine learning models, we did not have the specialized expertise required to modify cutting-edge models or to develop them from the ground up. Consequently, the work leveraged two pre-existing models —Bert-Large-NER for identifying characters, and GPT-4 for generating character summaries —and tuned them using techniques not requiring forefront expertise:

- **BERT-Large-NER**: As further discussed in section 3.2.1, a limitation of BERT-Base-NER is its occasional failure to fully classify all components of a new entity, leading to partial recognition where only some tokens of the entity are identified. This results in fragmented entities comprised of accurately classified, yet disjointed, tokens. This work proposed a strategy to reconstruct the complete entity from these fragmented pieces, explored in section 4.1.1.5.

- **GPT-4**: The tuning of GPT-4 for optimal output involved a two-pronged approach:

    1. An initial draft of the prompt template was created, guiding GPT-4 in summarizing characters. This template, defined in section 3.2.2.2, was inspired by prior research 2.3 that identifies common elements in fictional summaries, reviewed in section 2.3.2.
    2. Utilizing this template, character summaries were generated from a range of E-books, followed by iterative refinements to enhance the template's accuracy and consistency.

The rationale for selecting these particular models was as follows:

- **Bert-Large-NER**: BERT-Large-NER is ranked as one of the top-trending token classification models on HuggingFace —a platform for sharing machine learning models —as of April 9, 2024. Although other models, such as GLiNeR [10], which allows custom label definitions, rank higher, their popularity appears to be driven by their flexibility rather than superior performance. Comparative testing revealed that Bert-Large-NER outperformed GLiNeR in identifying people.

- **GPT-4**: GPT-4 was selected for the character summarization for two reasons:

    – The model's performance met the needs of this work. As explained in chapter 3, the E-books must be segmented into chunks manageable for LLMs without surpassing their context window limitations. GPT-4's context window can handle up to 8192 tokens, accommodating full or substantial portions of chapters within a single prompt.

    – While using GPT-4 incurs a cost (approximately $30 for 1 million tokens), it remained a viable option compared to open-source alternatives for several reasons:

* GPT-4 is available on OpenAI's cloud services, eliminating the need for us to provide computational resources.
* BookBeat had granted us access to the company's GPT-4 API keys, with reasonable token usage limits. BookBeat also offered a backup option of computational resources (a computer with 16 GB RAM) if a transition to open-source LLMs became necessary. However, given the significant performance reduction that would result from using open-source LLMs, GPT-4 remained the preferred choice. For instance, Meta's Llama 2 only allows up to 4096 tokens, half of GPT-4's capacity. Moreover, the computer provided by BookBeat would likely only support the smallest version of Llama 2, which contains merely 7 billion parameters compared to GPT-4's 1.76 trillion parameters. This reduction in performance could confound the analysis of poorly crafted summaries, making it difficult to discern whether issues arose from the prompt template or the LLM itself.

1.6 Delimitations

The scope of the work was delineated into three distinct categories, which are outlined in the subsequent sections: tasks that were *within scope*, those that were *potentially within scope*, and tasks that were *out of scope*.

1.6.1 Within Scope

The following were within the scope of the work:

- Developing a feature for creating character summaries from partial or complete English-language e-books.
- Designing and implementing a user-friendly interface to present these character summaries to the users.
- Achieving acceptable performance in the character summarization task based on established metrics commonly used for evaluating the efficacy of text summarization techniques.

1.6.2 Potentially Within Scope

The work could potentially have been extended to include the following tasks:

- Crafting a feature for summarizing entities within the text other than characters, such as locations or events.
- Generating general summaries of E-books that capture the narrative up to a designated point in the plot.
- Developing an interface that allows users to access character summaries while listening to audio books.
- Refining the character summary generation process by employing more sophisticated evaluation metrics.

1.6.3 Out of Scope

The following items fell outside the scope of this thesis:

- Assessing the impact of the final product on the user's reading or listening experience.
- Expanding the features detailed in sections 1.6.1 and 1.6.2 to accommodate languages other than English.

1.7 Structure of the Thesis

The thesis has the following structure:

- Chapter 2 presents previous research within ttwo areas: the structure of character summaries, and summarization of lengthy text documents.
- Chapter 3 presents the methodology and method used to solve the problem.
- Chapter 4 demonstrates the implementation of the solution.
- Chapter 5 showcases the results of the work.
- Chapter 6 discusses the results.
- Chapter 7 contains reflections on the results and the potential for future research.

Chapter 2

Background

This chapter provides the requisite background knowledge and contextualizes the research within the existing body of work. It begins with an overview of concepts related to LLMs. Next, it proceeds with an exploration of text summarization from a broad perspective, and then narrows the focus to the summarization of fictional characters. Then, it reviews methodologies for summarizing lengthy documents, with an emphasis on a technique known as hierarchical merging. Lastly, it concludes with a critical examination of various evaluation metrics, highlighting their comparative strengths and weaknesses.

2.1 Large Language Models (LLM) and Related Fields

An LLM is a machine learning model that has been trained on vast amounts of data to make them capable of processing and generating natural language [11]. In this section, concepts related to such models are introduced. These concepts include LLM-specific terminology, concepts from related fields, as well as terms coined by us to facilitate discussions on the application of LLMs to character summarization.

2.1.1 Linguistics

Before delving into LLM-specific terminology, it is essential to review some linguistic concepts frequently referenced in LLM literature or that can facilitate upcoming discussions.

Linguistic Unit According to dictionaries [12], a linguistic unit is any elementary unit of language. In the context of text processing by LLMs, a linguistic unit commonly refers to a word, a part of a word (commonly called a *subword*) or a character [13].

Discourse Unit In linguistic literature [14], a discourse unit is formally defined as a "*text segment with linguistic properties which are used to construe both semantic representations (interpretations, inferences) and the text and context models at stake.*" In this thesis, it is specifically used to denote text segments such as sentences, paragraphs and chapters.

2.1.2 Prompt Engineering

This section defines concepts primarily based on [5], with exceptions noted as needed.

Prompt A *prompt* is a set of instructions given to the LLM to perform a desired task.

Prompt Components A prompt is composed of *prompt components*, relevant to this thesis being the *instructions* themselves, and an optional *context* that the LLM considers when producing its response.

Token A prompt and its response can be further dissected into their most basic units: *tokens*. In models processing text, a token is a linguistic unit tailored to the specific needs of the model.

Context Window The *context window* refers to the maximum size of the prompt that the LLM can process [15].

Chunk When the desired context exceeds the allowed context window, it must be divided into smaller *chunks*, a procedure referred to in this thesis as *chunkification*. Typically, the instructional component of a prompt is small, but the ideal context often requires chunkification due to its size.

2.1.3 Other Concepts

This section explains concepts that do not fall under any of sections 2.1.1 and 2.1.2 and are coined by us.

Cohesive and Incohesive Chunks A *cohesive chunk* is composed entirely of a certain type of discourse unit, whereas an *incohesive chunk* does not maintain such uniformity and may consist of incomplete discourse units, such as text ending mid-sentence.

2.2 Text Summarization

Text summarization is a process that involves three key elements: the *summary*, which is the end product; the *source text*, which serves as the starting material; and the *transformation* process that converts the source text into the summary. This section delineates these three elements, drawing on insights from existing literature [16], and explains their application within this work.

2.2.1 Summary

- **Definition**: Summaries are categorized as either *indicative* or *informative*. Indicative summaries capture the salient content of the source text, whereas informative summaries are a condensation of all content, effectively serving as a substitute for the original document.
- **Usage**: Since this work aimed to construct the summaries out of the components specified in section 3.2.2.2, rather than furnish a full portrayal, the generated summaries are more indicative than informative.

2.2.2 Source Text

- **Definition**: The source text is the material from which the summary is derived. In this work, the source text was selected from one of the chunks that together make up an E-book. An E-book can be viewed as being composed of two main parts: the *story*, typically divided into chapters; and the *paratext*, which includes supplemental materials such as tables of contents, prefaces, afterwords, acknowledgments, glossaries, and appendices.
- **Usage**: This work endeavored to concentrate on the story by excluding the paratext from the source text for summary creation, ensuring that the summaries are based solely on the story itself.

2.2.3 Transformation

- **Definition**: The transformation, i.e. the summarization, can be classified into two approaches: *extractive* and *abstractive*. Extractive summarization refers to the process of extracting salient sentences from the source text and compiling them into a summary. Abstractive summarization, on the other hand, involves creating a summary anew by interpreting the source text through an internal semantic framework, allowing for greater creativity and rephrasing.

- **Usage**: This work did not specify to the LLM which summarization technique to use. Rather, the model was instructed to focus on certain attributes when creating character summaries, as detailed in section 3.2.2.2. The LLM was thus given the autonomy to choose the summarization method it deems most appropriate.

  However, the discretion granted to the LLM in the choice of summarization technique was an important consideration when assessing the outcomes. Some evaluation metrics, like Recall-Oriented Understudy for Gisting Evaluation (ROUGE), assess the quality of a generated summary based on its *lexical* rather than *semantic* overlap with the gold-standard summary, also known as reference summary. This is further discussed in section 2.5.1.1. Such metrics may not be suitable if the summary is produced through abstractive summarization, which can maintain the original meaning of the source text even without any lexical overlap with the reference summary.

2.3 Character Summarization

To the best of our knowledge, the only scholarly work on the automatic generation of character summaries using machine learning is a master's thesis by Zhang from 2017 [16] and a subsequent paper by Zhang and colleagues in 2019 [17]. The latter does not significantly deviate from the former. Moreover, Zhang's thesis provides a more comprehensive explanation of his character summarization methodology. Therefore, this section will focus exclusively on the master's thesis by Zhang.

2.3.1 Primary Objective: Inference of Salient Attributes

Zhang's thesis primarily aims to design a model that can deduce salient attributes to create effective character summaries with. Attributes in this context refer to descriptions of characters consisting of a short sequence of words. This process begins with a regression model that selects attributes — $A^{selected}$ —based on their frequency within the story. The chosen attributes are then presented to the authors of the original books through surveys, asking them to identify which predicted attributes they consider representative of their characters, denoted as A^{target}. This set serves as the ground truth for evaluating the model's predictive accuracy. The model's performance is quantified using the ROUGE score $\frac{A^{selected} \cap A^{target}}{A^{target}}$, which measures the proportion of overlap between the model-selected attributes and the author-identified attributes.

2.3.2 Secondary Objective: Structure of Fictional Summaries

Zhang's research also explores the structure of character summaries by analyzing common elements in fictional story summaries from WattPad, a platform for sharing fiction. The analysis identifies six recurring components:

- **Description of main characters**: Typically at the start of the summary, this component introduces the main character.
- **Setting**: This element outlines the story's backdrop and may or may not be included in the summary.
- **Foundational event**: A pivotal event that sets the stage for the story, often positioned in the middle of the summary.
- **Hook**: A brief and engaging text designed to pique the reader's curiosity, usually not part of the story.
- **Conclusion**: The story's ending, which is rarely disclosed in the summary to avoid spoilers.

While these elements pertain to story summaries, Zhang suggests that the structure for character summaries can be extrapolated from the "description of main characters" section. The suggested format includes the character's name followed by their salient attributes. However, Zhang's framework lacks

guidance on organizing these traits. For instance, if relationships to other characters are regarded as a salient attribute, then there should be a natural location within the summary into which that relationship is to be inserted, just like there are natural locations for the setting or foundational event. Despite this, the structural elements of fictional story summaries informed the design of character summaries. Section 3.2.2.2 of this thesis further discusses the approach to structuring character summaries, drawing on Zhang's analysis.

2.3.3 Conclusions

Zhang's insights into the structure of fictional summaries are valuable; however, their approach to machine learning-based character summarization has limited applicability to this work. Zhang's methodology involves building a summarization model from the ground up, whereas this work utilized pre-existing LLMs for the task. Furthermore, Zhang's research predates the emergence of advanced LLMs, such as GPT-3.5, which was introduced in 2022 [18]. Consequently, their work does not incorporate the capabilities of current state-of-the-art models.

2.4 Hierarchical Merging

In order for LLMs to summarize book-length documents, the document must first be segmented into chunks that fit within the LLM's context window. Two strategies for this partitioning are discussed by Chang et. al: *hierarchical merging* and *incremental updating* [8]. While incremental updating should perform better than hierarchical merging in theory, according to the study, empirical evidence from benchmarks indicates that its superiority is marginal in terms of generating summaries without inconsistencies. Furthermore, it is inferior to hierarchical merging when considering other types of errors identified by Chang et al. Consequently, this work focused exclusively on hierarchical merging in section 3.2.2 and did not delve into incremental updating.

2.5 Evaluation Metrics

This section provides an overview of various metrics that are potentially relevant for the evaluation of the generated summaries. Based on this overview, the metrics were contrasted contrasted against each other and served

as a foundation for selecting evaluation method. The content of this section is primarily based on Eugene Yan's article, *Evaluation & Hallucination Detection for Abstractive Summaries* [19], with a few exceptions. When such exceptions occur in the text, they are accompanied with references to their source.

The evaluation metrics used when assessing summaries are commonly categorized as groups, the ones of relevance for this thesis being *reference-based* and *context-based* metrics. First, reference-based metrics are reviewed in section 2.5.1, followed by context-based metrics in section 2.5.2. Then, the comparison of the metrics is done in section 2.5.3. After having established reference-based BERTScore as the evaluation metric used to evaluate this work's results in section 2.5.3, some final remarks on how this work interpreted BERTScore's F1 score are presented in section 2.5.4.

2.5.1 Reference-based Metrics

Reference-based metrics are a class of metrics used for evaluation of text summarization models. It is the most common type of evaluation metrics for text summarization, and operates by comparing machine-generated summaries against reference summaries. Reference summaries are ideal summaries, which are typically human-crafted exemplars serving as benchmarks for quality and referred to in reference-based metrics terminology as "references".

2.5.1.1 ROUGE

ROUGE is a predominant subset of reference-based metrics. It quantifies the quality of a generated summary by measuring the overlap of *n*-grams between the summary and the reference, normalized by the total count of *n*-grams in the reference. Mathematically, this is expressed as:

$$\frac{\sum_{S \in Reference\ Summaries} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in Reference\ Summaries} \sum_{gram_n \in S} Count(gram_n)} \tag{2.1}$$

This fraction is referred to as "overlap ratio" in this thesis. The term *n*-grams refers to sequences of *n* contiguous linguistic units, such as words. ROUGE-N, for example, assesses the overlap of consecutive *n*-grams, with ROUGE-2 focusing on bi-grams, ROUGE-3 on tri-grams, and so forth.

To address the limitations of consecutive *n*-gram overlap, variants like ROUGE-S allow for non-consecutive *n*-grams to be considered overlapping as long as they appear in the same order.

2.5.1.2 BERTScore

ROUGE-N and ROUGE-S are not suitable for evaluating summaries generated using abstractive summarization, as the LLM might employ synonyms or rephrase content as discussed in section 2.2.3, leading to semantic albeit not lexical overlap with the reference. In these cases, BERTScore emerge as a more appropriate metric.

In this section, BERTScore is defined by first examining *cosine similarity*, a fundamental concept for understanding the metric. Next, the traditional definitions of the performance metrics *precision*, *recall* and *F1 score* are reviewed to lay the groundwork for understanding BERTScore's definitions of said concepts.

Cosine Similarity BERTScore evaluates summaries by converting them and their references into *vector embeddings*, which represent the semantics of the text as vectors. It then calculates the cosine similarity between these embeddings. Cosine similarity, as defined in Data Mining texts [20], measures the similarity between two vectors, x and y, as follows:

$$sim(x, y) = \frac{x \cdot y}{\|x\| \times \|y\|} \tag{2.2}$$

By measuring cosine similarity, BERTScore gauges the semantic equivalence between the generated summary and the reference, accounting for synonymous expressions that ROUGE cannot detect. Specifically, BERTScore applies cosine similarity to general machine learning metrics known as precision, recall and F1 score to formulate its own metrics. To understand how cosine similarity is applied, the traditional definitions of precision, recall and F1 score are examined next.

Traditional Precision, Recall and F1 Score The concepts' definitions presented below are borrowed from [21]. Moreover, the accompanying explanations are adapted to the context of this work to facilitate understanding.

Precision measures the accuracy of the information present in the generated summary. High precision means that the generated summary includes information that is mostly correct and relevant.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{2.3}$$

Recall measures the completeness of the generated summary from by calculating the proportion of relevant information that has been successfully

retrieved. High recall means that the generated summary covers most of the important points from the reference summary.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{2.4}$$

The F1 Score measures the mean of the precision and recall, providing a single metric that balances both. The F1 score is useful when a balance between precision and recall is to be found, as it ensures that it is both accurate (precision) and comprehensive (recall).

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.5}$$

BERTScore's Precision, Recall and F1 Score The paper that first introduced BERTScore [22] redefines precision, recall and F1 in terms of cosine similarity as follows:

Let $x = \{x_1, x_2, ..., x_i\}$ denote references and $\hat{x} = \{\hat{x}_1, \hat{x}_2, ..., \hat{x}_j\}$ be the "candidate", i.e. text that is evaluated against the references, where x_i and x_j refer to the tokens of the respective texts. Then, precision, recall and F1 score can be redefined as:

$$P_{BERT} = \frac{1}{\|\hat{x}\|} \sum_{\hat{x}_j \in \hat{x}} max_{x_i \in x} x_i^T \hat{x}_j \tag{2.6}$$

$$R_{BERT} = \frac{1}{\|x\|} \sum_{x_i \in x} max_{\hat{x}_j \in \hat{x}} x_i^T \hat{x}_j \tag{2.7}$$

$$F_{BERT} = 2 \times \frac{P_{BERT} \times R_{BERT}}{P_{BERT} + R_{BERT}} \tag{2.8}$$

The intuition behind BERTScore's adaptations of precision, recall, and F1 score mirrors that of their traditional counterparts, albeit with the addition of cosine similarity:

In P_{BERT}, $\frac{1}{\|\hat{x}\|}$ corresponds to the denominator in equation 2.2, and $\sum_{\hat{x}_j \in \hat{x}} max_{x_i \in x} x_i^T \hat{x}_j$ corresponds to the numerator, representing cosine similarity. Additionally, the latter expression also represents the numerator of equation 2.3, as the summation of each candidate token x_i that maximizes the similarity with $\hat{x}_j$ essentially captures the true positives. Similarly, the denominator in equation 2.3 is represented by the former expression since the part of $\hat{x}$ that overlaps with x constitutes the true positives, while the surplus that does not overlap accounts for the false positives.

Conversely, R_{BERT} is analogous to P_{BERT} but with x replaced by $\hat{x}$ and vice versa, thus it can be similarly justified as representing recall. F_{BERT}, corresponding to the F1 score, follows naturally from these definitions.

2.5.1.3 MoverScore

MoverScore operates similarly to BERTScore but extends the comparison to allow for many-to-one matches between tokens, offering a more flexible evaluation compared to BERTScore's one-to-one matching schema [19].

2.5.2 Context-based Metrics

Reference-based metrics are effective when abundant high-quality references are available for evaluation. However, this criterion posed a challenge for this work on two fronts: Firstly, while numerous character summaries exist, finding those based on chapters or other cohesive chunks is difficult but preferable for evaluating the summaries generated by this work, as they align with the goal of updating character summaries incrementally as the story unfolds to avoid spoilers. Secondly, although we could have crafted reference summaries manually by reading the source text, this was not feasible given the timeline of this work and our lack of expertise in literary summarization.

Therefore, context-based metrics, which do not rely on references, could be a better contender than reference-based metrics under certain circumstances. Like reference-based metrics, context-based metrics calculate an overlap ratio, but with respect to the context instead of a reference. Moreover, there are context-based adaptions of ROUGE, BERTScore and MoverScore [19], all which operate the same way as their reference-based counterpart but where the reference has been replaced by the context.

However, a drawback of the context-based approach manifests as the context grows large. In context-based BERTScore, for instance, larger contexts may dilute the overlap ratio when compared to concise summaries, given that the former contains multiple *concepts* and the latter few. In our understanding of the word, a *concept* as it is used here is some information that can be distilled into less text and still retain its original meaning. That is to say, if both the context and summary are embedded as vectors in the same space, the concepts in the context might blend while those in the summary remain distinct. Consequently, the similarity between the summary and the relevant section of the source text could be obscured. Hence, it is crucial to segment the source text into appropriately sized chunks to facilitate effective evaluation by context-based BERTScore.

2.5.3 Comparison

Out of the evaluation metrics reviewed in the subsections to section 2.5, reference-based BERTScore and MoverScore emerged as the most promising to use for this work for two reasons. First, context-based metrics could not be used since the chunks serving as context were large (roughly 8000 tokens) and the summaries small (max 400 tokens). If they were used, they would result in unreliable accuracy scores as explained in section 2.5.2. Second, the only alternative, ROGUE, is unsuitable for abstractive summarization, which is a summarization technique likely used by LLMs.

We refrain from advocating for either of the two remaining options, reference-based BERTScore and MoverScore, as their distinction lies primarily in scoring stringency or leniency. Nonetheless, reference-based BERTScore was chosen as evaluation method for this work due to its seemingly wider adoption compared to MoverScore.

2.5.4 Interpretation of Reference-based BERTScore's F1 Score

To paraphrase the paragraph "Traditional Precision, Recall and F1 Score" in section 2.5.1.2, precision measures the correctness of the retrieved information, while recall measures the coverage of relevant information. The F1 score combines both metrics. Moreover, the F1 score ranges from 0 to 1 [23], where 0 indicates no similarity between the generated summary and the reference summary, and 1 indicates they are identical. To the best of our knowledge, there is no consensus within the scientific community on what constitutes a good F1 score, other than the general understanding that closer to 1 is better. One blog post [24] suggests that an F1 score above 0.5 is acceptable and above 0.8 is good. However, another blog post [25] suggests that anything above 0.7 is good, contradicting the previous post. Neither statement is supported by arguments nor references.

Nonetheless, as [25] suggests, the definition of a good F1 score should depend on the context. For example, a medical diagnosis requires higher F1 scores than a character summary. Additionally, BookBeat, the company for which this work is conducted, had not specified performance expectations and views this work as an experimental investigation into the feasibility of the proposed functionality. Therefore, we could define what constitutes a good F1 score for this work based on our own standards, given that the definition aligns with the problem statement: "How can machine learning summarize

characters within English E-books in a spoiler-free manner?" Reasonably, such a definition should ensure that the results of this work provide valuable direction for future research. Put differently, the results should be more relevant than they are irrelevant, which in our minds translates to achieving an F1 score higher than 0.5, indicating that the generated summaries' overall recall and precision —the summaries' relevance and the correctness of the content that is relevant —is above average. However, while this threshold should be the minimum requirement, we set 0.6 as the desired threshold, to aim for a quality level that is more good than bad. This reasoning underpins the evaluation thresholds defined in section 1.4.

2.6 Summary

This chapter reviewed related work within four areas: text summarization from a general as well as a character-specific perspective, techniques for summarization of book-length documents, and metrics for evaluating text summarization. These are the conclusions of each respective section:

Text summarization was stated to be constituted by three components: a summary, a source text and a transformation which the source text undergoes to produce the summary.

Research on automatic summarization of characters was restricted to only one study which is partially outdated by today's standards. However, some inspiration in regards to how to structure character summaries could be drawn.

While there are other text summarization techniques, such as incremental updating, hierarchical merging appeared to be the most promising technique.

Lastly, two categories of evaluation metrics were examined: reference-based and context-based metrics. Out of the reviewed metrics, reference-based BERTScore emerged as the most promising candidate.

Chapter 3

Methodology

The purpose of this chapter is to provide an overview of the research method used in this work. Section 3.1 describes the research process. Section 3.2 elaborates on the methods used for generating the character summaries. Section 3.3 details the research paradigm. Section 3.4 focuses on the data collection techniques used for this research. Section 3.5 describes the experimental design. Section 3.6 describes the method used for the data analysis. Section 3.7 describes the framework that was selected to evaluate the generated character summaries. Finally, section 3.8 describes how documentation of the work was handled.

3.1 Research Process

This section delineates the methodologies used in this work to achieve its goals.

The final product of this work consisted of three services: a character summarization service, a server facilitating client access to summaries, and a client interface. Figure 3.1 illustrates how the services were interconnected, with `ML Program` generating character summaries, `Backend` serving as the server and `Web Application` as the client, and `DB` referring to .CSV files that contained character summaries of a given book.

The service which produced the character summaries was developed first, and then the server and client. This ordering minimized the risk of not attaining the goals in section 1.4, since they were not contingent upon the server and the client.

The detailed research process for each service unfolds as follows:

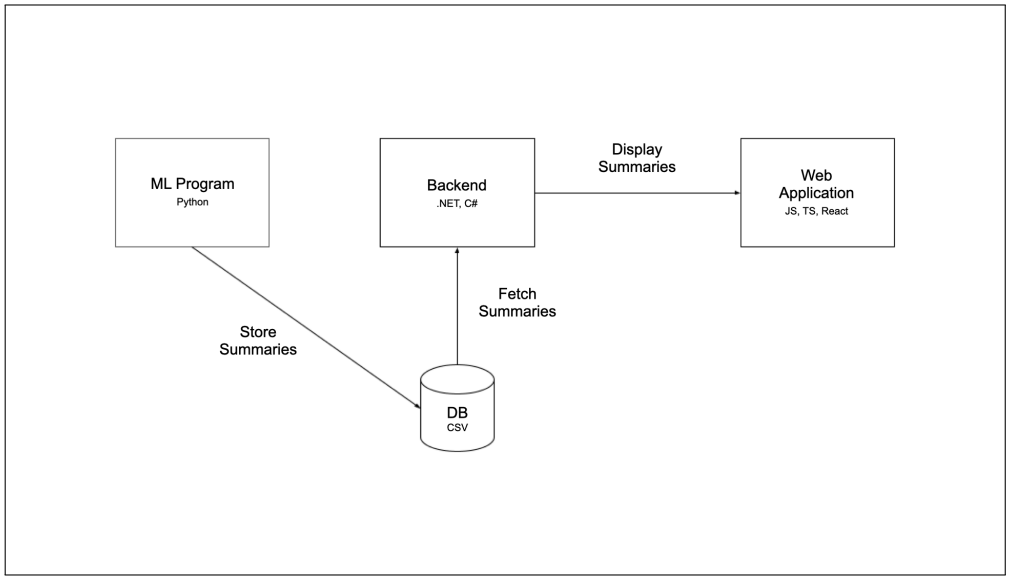


Figure 3.1: Overview

3.1.1 Character Summarization Service

The character summarization service was implemented in Python by doing the following:

1. Developing a function to pre-process the Electronic Publication (ePub) file —an E-book file format —extracting the story text by removing paratext.
2. Creating a function to parse the book contents into plain text from the pre-processed ePub files.
3. Utilizing the BERT-Large-NER model to extract a list of person entities from the parsed ePub file.
4. Refining the entity list from step 2, eliminating non-character entities, duplicates, and variations.
5. Segmenting the parsed ePub file into cohesive text chunks for consumption by the summarization model.
6. Designing an algorithm for summarization per text chunk, following hierarchical merging procedures (detailed in section 3.2.2), and store the summaries as .CSV files accessible to the server.

3.1.2 Server

The server was implemented using .NET by doing the following:

1. Implementing RESTful endpoints for client interaction.
2. Developing a function to retrieve character summaries from the stored .CSV files.

3.1.3 Client

The client is implemented using React by doing the following:

1. Creating mockups for client views, including:
    - A login page.
    - Book selection interface.
    - An E-book reader, which indicates to the user which characters that can be tapped in order to display its summary.
    - A panel which displays the character summaries to the user.
2. Implementing the client views.
3. Integrating the client with the server.

3.2 Summarization Methods

The character summarization entailed two primary steps: First, the extraction of character names from the E-book, followed by the integration of these names into pre-defined summarization prompts, termed *prompt templates* in this thesis. This section first delves into the method for character name extraction, followed by a review of hierarchical merging, which constituted the character summarization process.

3.2.1 Character Extraction

The names of the characters were extracted in five steps, illustrated in figure 3.2:

1. **Pre-processing**: The ePub file containing the E-book underwent pre-processing, involving the removal of paratext to mitigate the risk of extracting non-character entities, such as real people mentioned in forewords or acknowledgments sections, and incorrectly treating them as characters.

2. **ePub parsing**: The pre-processed ePub file was parsed into plain text.

3. **Chunkification**: The text was segmented into appropriately sized chunks to fit within the context window of the LLM.

4. **NER**: The chunks were scanned using the BERT-Large-NER model to identify entities classifiable as people.

5. **Post-processing and Filtering**: Identified entities underwent post-processing and filtering to retain only characters. Post-processing and filtering are defined as the following:

    - *Post-processing*: BERT-Large-NER occasionally fails in identifying all tokens of an entity, resulting in fragmented entities consisting of correctly classified, yet disjointed, tokens. To handle those scenarios, the letters to the left and right of those tokens were appended to the tokens as long as the letters were in fact letters and not spaces, punctuation or other special characters.
    - *Filtering*: A number of filters were applied to the post-processed entities:
        - Removal of any occurrences of an entity only consisting of a character's first or last name, given that there are multiple characters with the same first or last name, so that ambiguity is avoided.
        - Removal of any occurrences of an entity which is the plural form of a character's surname (e.g., the Cratchits).
        - Removal of prefixes, such as titles (Mr., Ms., Sgt.) or other honorifics (Baron, Father, Saint).
        - Removal of any occurrences of the author's name.

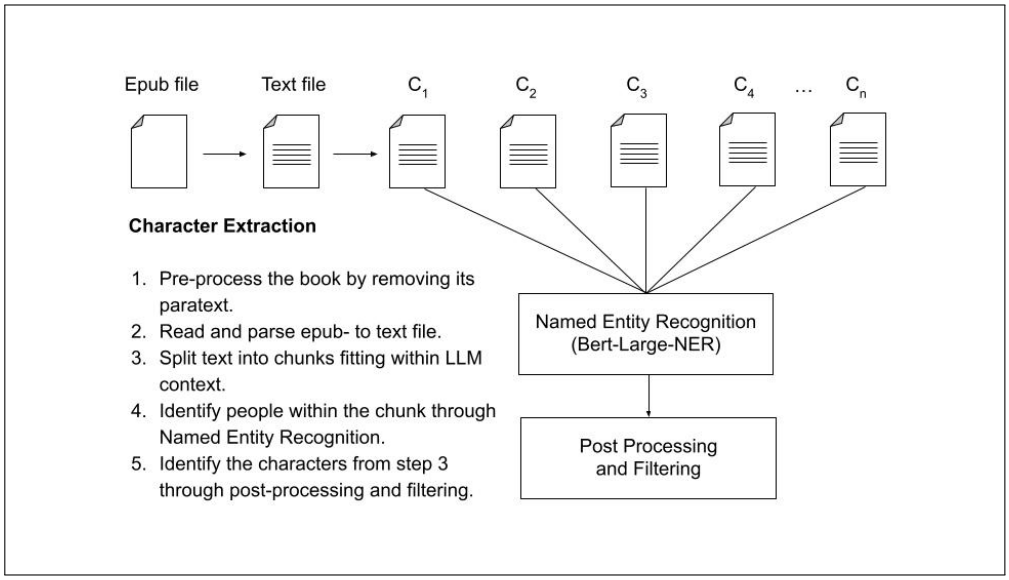


Figure 3.2: Character Extraction

3.2.2 Hierarchical Merging

Hierarchical merging refers to an algorithm used in this work to merge partial summaries generated from book chunks by repeatedly instructing an LLM to summarize the chunks and then merge them. The summarization and merging process was applied to five characters simultaneously to reduce the monetary costs incurred by iterating over the book.

While this section primarily concerns the algorithm, it also reviews the format of the prompt sent to the LLM. Therefore, first the algorithm is examined in section 3.2.2.1 and then the format in section 3.2.2.2.

3.2.2.1 Algorithm

Once the names of the character were extracted, the process of generating character summaries unfolded in four steps, as depicted in figure 3.3. Among the four steps outlined below, two steps start with the bolded text ”Iteration 1” and ”Iteration 2”, respectively. This notation simply emphasizes that book was iterated over twice when processed by the LLM: once to generate *subsummaries*, which in this work referred to summaries only based on one chunk, as opposed to *summaries*, which referred to summaries based on the $n-1$:th summary and n:th subsummary.

The steps for the algorithm are as follows:

1. The E-book underwent segmentation into cohesive chunks.
2. The first summary was based on the initial chunk.
3. **Iteration 1**: Subsequent chunks were summarized into subsummaries.
4. **Iteration 2**: Each subsummary was merged with its preceding summary into the final summaries.

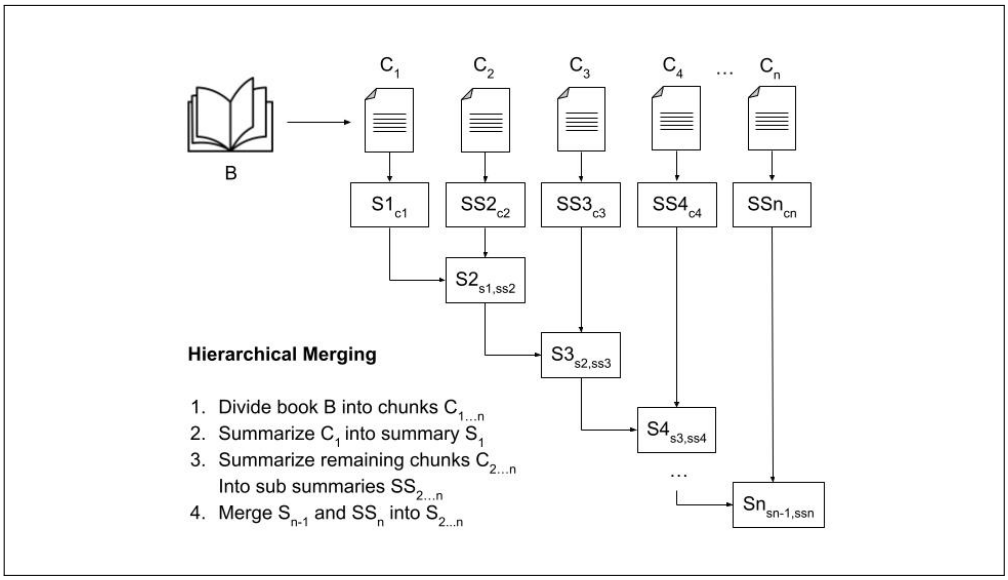


Figure 3.3: Hierarchical Merging

Initially, the chunks were intended to be iterated only once, merging the $n-1$:th summaries with n:th chunks to form the final summaries. However, the actual implementation involved two iterations. This deviation arose because the actual implementation was conceived in the early stages of the work and executed accordingly. Conversely, the planned implementation, which likely offers greater efficiency in terms of both time and monetary costs, was formulated in the later stages but demanded time that we lacked at that juncture.

Further explanation of the planned implementation can be found in appendix C.

3.2.2.2 Format

To generate a character summary, the LLM was instructed to adhere to a pre-defined structure, referred to as the *format* in this thesis. This structure,

informed by our subjective preferences, invites interpretation and discussion on its validity, a topic discussed further in section 3.3.

The format contains three components, which we have coined ”character components” to facilitate their discussion within this thesis: (i) the character's relations to other characters, (ii) their personality traits, and (iii) their appearance. Furthermore, in the same vein as Zhang's analysis of fictional summaries, reviewed in section 2.3.2, an order was imposed upon the components: Specifically, they were ordered in the same order they were listed above. Additionally, the character's name was placed at the front of the summary before any mention of the character's relations to other characters. Lastly, the LLM was tasked with generating summaries within a maximum constraint of 400 tokens, approximately 300 words.

3.3 Research Paradigm

This work investigated the realm of machine learning applied to the summarization of fictional characters through a dual lens: interpretivistic and quantitative, borrowing their definitions from [26]. Put simply, it operated under the premise that while the quality of these summaries remains open to interpretation, their effectiveness can be quantified based on pre-defined criteria. In essence, this work adopted a interpretivistic stance by proposing a subjectively ideal structure for the character summaries and delineating essential components. Simultaneously, it employed a quantitative approach by leveraging reference-based metrics to objectively evaluate the quality of these summaries.

Moreover, the work adopted a deductive approach in structuring character summaries, wherein we employed reasoning to determine the format. However, the data was collected inductively from existing resources of E-books.

3.4 Data Collection

The E-books that served as the source texts for the character summarization were collected from Project Gutenberg, a website hosting more than 70 000 E-books within the public domain. While we had access to the E-books maintained by BookBeat, the majority of them could not be used for this thesis since the copyright holders prohibit manipulation of the books through the usage of generative AI. Therefore, the thesis only collected its data from books

residing within the public domain.

3.4.1 Sampled Books

The books *Animal Farm* by George Orwell, *The Odyssey* by Homer and *War and Peace* by Leo Tolstoy were selected as evaluation samples. Increasing the number of books would raise the monetary costs incurred by GPT-4; therefore, three books were deemed sufficient.

The selection of books was based on two criteria:

- There should be a balance between short books (<100 pages), medium-length books (300 pages), and long books (>1000 pages). Dividing the data into these three sets ensures that biases due to the number of summaries that are merged by hierarchical merging are exposed.
- As discussed in section 3.7, only hierarchical merge was evaluated. Therefore, characters that are easily identifiable by BERT-Large-NER were selected, so that only the performance of hierarchical merge was measured.

The page count for each book was estimated by calculating the number of tokens in the book, converting the number of tokens into words based on how many tokens a word tend to be in the English language [27], and lastly converting the number of words into pages. This conversion was done by calculating the average number of words per page of the Penguin Classics translation of *War and Peace* [28], the longest sample book, and dividing the word count of each book by that number. In other words, if $\hat{f}_{page\ count}(book_{word\ count})$ denotes the estimated page count as a function of the word count, then:

$$words\ per\ page_{avg} = \frac{War\ and\ Peace_{word\ count}}{War\ and\ Peace_{page\ count}}$$

$$= \frac{War\ and\ Peace_{token\ count}}{War\ and\ Peace_{page\ count} * 1.3} = \frac{766229}{1440 * 1.3} = 409$$

$$\implies \hat{f}_{page\ count}(book_{word\ count}) = \frac{book_{word\ count}}{words\ per\ page_{avg}}$$

Book	# Tokens	# Words	# Pages
Animal Farm	36939	28415	69
The Odyssey	165959	127661	312
War & Peace	766229	589407	1440

Table 3.1: Length of sample books

3.4.2 Sampled Characters

The characters from each book in section 3.4.1 were ranked by their number of occurrences within the text, as can be seen via the links listed in section A.1 of appendix A. From each book, the three most frequently mentioned characters and the three least frequently mentioned characters were selected. However, if a book had more than 20 characters, the characters ranked 17th to 20th were selected instead of the least frequently mentioned ones. This adjustment was made because we had summarized only the first 20 characters from each book to keep the monetary costs down. In total, 18 characters were sampled: 9 of them were the most frequently mentioned characters, as shown in table 3.2, and 9 were less frequently mentioned, as shown in table 3.3.

Book	Character	Frequency
Animal Farm	Napoleon	160
	Snowball	99
	Boxer	67
The Odyssey	Ulysses	632
	Telemachus	272
	Jove	162
War and Peace	Pierre	1925
	Natásha	1188
	Rostóv	729

Table 3.2: The most frequently occurring characters in the sample books.

Book	Character	Frequency
Animal Farm	Jessie	4
	Minimus	3
	Pincher	2
The Odyssey	Aegisthus	29
	Eurymachus	28
	Helen	26
War and Peace	Bourienne	124
	Alexander	122
	Rostopchín	116

Table 3.3: Less frequently occurring characters in the sample books.

Although this grouping of less frequently occurring characters may seem misleading, it should not have impacted the overall analysis of the results in later chapters. For instance, the ratio between the frequency of the most mentioned character within the group of least mentioned characters for a specific book, denoted as $maxarg_{character \in least\ frequent_{book}}$, and the most mentioned character within the group of most frequently mentioned characters, denoted as $maxarg_{character \in most\ frequent_{book}}$, did not differ significantly, as shown below:

$$\frac{maxarg_{character \in least\ frequent_{Animal\ Farm}}}{maxarg_{character \in most\ frequent_{Animal\ Farm}}} = \frac{4}{160} = 0.025$$

$$\frac{maxarg_{character \in least\ frequent_{The\ Odyssey}}}{maxarg_{character \in most\ frequent_{The\ Odyssey}}} = \frac{29}{632} = 0.045$$

$$\frac{maxarg_{character \in least\ frequent_{War\ and\ Peace}}}{maxarg_{character \in most\ frequent_{War\ and\ Peace}}} = \frac{124}{1925} = 0.064$$

Given that all these ratios are fairly similar, with the smallest and largest ratios differing by only about 0.04, the characters in table 5.2 should have been representative of the same frequency stratum.

3.5 Experimental design and Planned Measurements

In order to reproduce the environment in which the evaluation is performed, the following must be arranged:

- The dependencies in the link listed in paragraph A.2.0.0.2 of appendix A must be installed.
- There are no requirements on the hardware, but we recommend running the machine learning service for producing the character summaries on a computer with at least 8GB RAM.

3.6 Data Analysis

While numerous aspects of this work were worthy of analysis, the primary focus of the planned data analysis was centered on the performance of hierarchical merging. This focus was warranted as hierarchical merging served as the linchpin of the thesis, with all other functionalities developed to complement it. In this thesis, the latter is referred to as the *peripheral functionalities*.

During analysis, the terms *candidates* and *references* was used instead of *generated summaries* and *reference summaries* respectively for brevity, as these concepts are frequently referred to. The analysis of hierarchical merging focused on two facets: content adherence and format adherence.

3.6.1 Analysis of Content Adherence

Content adherence refers to the degree of semantic similarity between the candidates and the references. Prior to the analysis, the semantic similarity was measured by evaluating the candidate's BERTScore, a procedure delineated in section 3.7.1. After obtaining the BERTScore results, they were analyzed to assess reliability and validity [29]: On the one hand, the reliability was quantified by computing the confidence interval for the F1 score. On the other hand, the validity was determined by deriving the mean of the the F1 score. The confidence interval is computed as $CI = \overline{x} \pm q^N_{0.95} \times \frac{s}{\sqrt{n}}$, where:

- The F1 score for each respective candidate $x = \{x_1, x_2, ..., x_n\}$.
- The mean F1 score $\overline{x} = \frac{\sum_{i=1}^{n} x_i}{n}$.
- The 95th percentile of the normal distribution $q^N_{0.95} = 1.96$.
- The sample standard deviation $s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n-1}}$.
- The total number of candidates $n = 9$ for the most frequently and less frequently mentioned characters respectively.

3.6.2 Analysis of Format Adherence

The term *format* is used in this thesis to refer to the summary structure defined in section 3.2.2.2, which was implemented as a combination of two elements:

- A prompt template, which instructed the LLM on the character components to include in the summary. The implementation details are found in the subparagraph ”Prompt Template #1: Subsummarization” in section 4.1.2.2.
- A prompt example, which guided the LLM on the order in which these components should be arranged within the summary. Its implementation is detailed in the paragraph ”Prompt Examples” in section 4.1.2.2.

Format adherence, therefore, refers to the degree of adherence to (i) the order of the components within the format, and (ii) the number of such components included in the summary. Before the analysis commences, the order of the components within the candidates and references were scrutinized respectively by counting the number of components that aligned with the format, a procedure detailed in section 3.7.2. The results were then analyzed to assess reliability only. The standard deviations and means for the parameters listed in 3.7.2 were derived to indicate the reliability in terms of format adherence. Although numerical values are obtained for the reliability, they were merely used as points of references that underpinned a qualitative analysis of the format adherence, and were not measured against some numerical threshold.

3.7 Evaluation Framework

Although various benchmarks exist for gauging the the degree of completion of the peripheral functionalities, we did not have sufficient time by the end of the thesis to construct such benchmarks. Therefore, only hierarchical merging was evaluated, first with respect to its content adherence, and then its format adherence, concepts that are defined in section 3.6.

Like section 3.6, the terms *candidates* and *references* were used instead of *generated summaries* and *reference summaries* respectively for brevity.

3.7.1 Evaluation of Content Adherence

To explain the evaluation of the content adherence, it is essential to disclose three underlying assumptions.

- The evaluation of GPT-4's summarization capabilities was considered irrelevant for this thesis, as such assessments have been previously conducted [30].
- The summary derived from the final subsummary and its antecedent summary was presumed to be the most critical for evaluation. This assumption stemmed from the notion that if any distortion of content occurs during the merging of a subsummary and its preceding summary, the last summary should exhibit the most pronounced distortion.
- Since BERTScore was used for the evaluation, both the candidate and the reference had to adhere to the summary structure outlined in section 3.2.2.2. Failure to meet this structural alignment may have resulted in an inaccurate assessment, potentially penalizing the candidate despite its quality, solely due to differing focal points between the candidate and the reference.

Given these assumptions, the evaluation framework for the content adherence was formulated as as a two-step procedure:

1. Provide ChatGPT-4 with prompt #1 —as described in the subparagraph "Prompt #1: Subsummarization" in section 4.1.2.2 —to generate a reference summary based on its innate knowledge of public domain books.
2. Calculate the precision, recall and F1 score —according to BERTScore's definitions of the concepts, laid out in section 2.5.1.2 —for the candidate relative to the reference generated in the prior step.

3.7.2 Evaluation of Format Adherence

The evaluation framework for the format adherence can be expressed as a two-step procedure:

1. Text segments within both the candidates and the references were annotated with numbers from (1) to (3) to indicate that the segment corresponds to one of the three character components. Specifically,

(1) referred to a character's relations to other characters, (2) denoted personality traits, and (3) corresponded to the character's appearance. Correctly ordered components were highlighted with green, whereas all other components were highlighted with red. If multiple text segments could be identified as one of the components, then only the first segment was annotated. Furthermore, any negations of the components —e.g., "There is no mentions of his or her appearance" —were annotated as well. The unannotated and annotated versions of the summaries are found in paragraph A.2.0.0.3 of appendix A.

2. Four metrics were derived from the annotated appendix:

   - $cand_{ordered}$: the total number of components that were correctly located within the candidates.
   - $cand_{total}$: the total number of components that were present within the candidates.
   - $ref_{ordered}$: the total number of components that were correctly located within the references.
   - ref_{total}: the total number of components that were present within the references.

3.8 System Documentation

The work was thoroughly documented within a Github repository, accessible to stakeholders. Within this repository, all essential programs and code needed to execute them were included, such as software for character extraction, summarization, the server, and the client prototype. Moreover, the codebase featured a README file containing instructions for setup and execution, along with a list of required libraries. Additionally, the repository hosted this thesis for further insight and reference.

Chapter 4

Implementation

In this chapter, the implementation of the character summarization service, the server and finally the client are reviewed.

4.1 Character Summarization Service

The work that the character summarization service is responsible for can be divided into a two-step process: *character extraction* and *character summarization*. In other words, first the characters of the subject book had to be determined. Then, the characters were summarized. These two steps are reviewed in sections 4.1.1 and 4.1.2 respectively.

4.1.1 Character Extraction

The primary step of character extraction was the Named Entity Recognition, a process detailed in section 4.1.1.4, executed by the model BERT-Large-NER discussed in Section 1.5. However, prior to inference, several preparatory steps were necessary to refine the model's input and ensure its processing capabilities. Additionally, post-processing and filtering were essential to address imperfections in the BERT-Large-NER output. We elaborate on these steps in section 3.2.1,

4.1.1.1 Pre-Processing

To remove the paratext and thus minimizing the risk of extracting non-character people mentioned within the paratext, a list of words frequently recurring in paratext headings was used to locate sections that are likely to belong to the paratext and delete them.

The words were chosen in a two-step procedure: First, we asked ChatGPT-4 to generate a list of common paratext headings. Then, words that we deemed likely to also occur in non-paratext headings —e.g. the word ”License” could potentially be part of a chapter heading such as ”License to kill” —were removed from the list. Specifically, the list of words in listing B.1 in appendix B was used.

4.1.1.2 ePub Parsing

Given that the structure of an ePub file relies on XHTML, which can complicate content manipulation, a custom parser developed by BookBeat was utilized to strip away all XHTML elements from the file. The parser can be viewed from the link listed in paragraph A.2.0.0.4 of appendix A.

4.1.1.3 Chunkification

Since BERT-Large-NER expects a maximum of 512 tokens as input, and E-books generally consist of tens if not hundreds of thousands tokens, the source text had to be chunkified. However, unlike the chunks generated for the summarization step, we deemed it sufficient to generate incohesive chunks for the character extraction, i.e. chunks that may contain incomplete discourse units as elaborated in section 2.1.3. While the character extraction may have benefited from cohesive chunks, since BERT-Large-NER is then given a more complete context to infer the entities from, we did not notice any notable misclassifications from providing the model with incohesive chunks. Therefore, the source text was chunkified into equally sized and incohesive chunks using the LLM library LangChain.

LangChain provides various utilities for handling LLM related tasks, one being text splitters, such as the function `from_huggingface_tokenizer()` from the module `RecursiveCharacterTextSplitter`. This function primarily accepts two arguments: a HuggingFace `tokenizer`, i.e. an object containing methods for processing tokens according to a HuggingFace model's definition of a token; and `chunk_size`, i.e. the desired size of the chunks into which the text is to be splitted, measured in tokens. It may also take a `chunk_overlap` as an argument, which specifies the number of tokens each chunk may have in common and thus allowing for an overlap between chunks.

Listing B.2 in appendix B demonstrates our implementation of the function for chunkifying the source text passed to BERT-Large-NER, using said LangChain utilities. Here, the BERT-Large-NER tokenizer and the

maximum chunk size allowed by the model, 512, were passed as arguments. Furthermore, the chunk overlap was set to 0 and thus disallowed. Finally, the source text was chunkified by invoking the function `split_text` of the object returned by `from_huggingface_tokenizer()`.

4.1.1.4 Named Entity Recognition (NER)

Once BERT-Large-NER had been passed the chunks generated from the previous step, it could proceed with inferring people from the book. This was accomplished by using so-called pipelines from the HuggingFace library *transformers*, which are objects serving as APIs between the user and the model.

As seen for the function `extract_character_names()` in listing B.3 in appendix B, a pipeline labeled `nlp()` was created by passing to the function `pipeline()` the string `"ner"`, specifying NER as the desired type of classification; an instance of BERT-Large-NER labeled `bert_model`; and a tokenizer for the model. Then, the chunks were iterated over and passed to `nlp()` for the inference of people in the book. Note that the post-processing logic was also contained within `extract_character_names()`, which is demonstrated in the next step. Therefore, the post-processing logic is omitted from listing B.3 and instead denoted as "...Post-processing..." for brevity.

4.1.1.5 Post-Processing

This step is about processing the output from BERT-Large-NER after the inference due to the output having imperfections that makes it largely unusable for the subsequent character summarization. In this section, the imperfections are first reviewed. Then, the post-processing measures are examined.

Imperfections The output of BERT-Large-NER had two imperfections that required post-processing:

- *Incorrect classification of B-PER and I-PER subwords*: An entity inferred by BERT-Large-NER is structured as a JSON object consisting of a number of fields, the most relevant for our purposes being the `entity` field. The `entity` field represents the entity class, and after the inference its value is set to one of five class abbreviations predefined by BERT-Large-NER: (i) *O*, outside of a named entity; (ii) *MISC*, a miscellaneous entity; (iii) *PER*, a person's name; (iv) *ORG*,

an organization; and (v) *LOC*, a location. Since characters for the most part are people, only PER entities were of relevance for this work.

On a sidenote unrelated to the imperfection at hand, it should be noted that characters are not always people. In some books, such as *Alice's Adventures in Wonderland* by Lewis Caroll, the majority of the characters are inanimate objects such as playing cards, or animals such as The Cheshire Cat or The Caterpillar. In such cases, BERT-Large-NER is unable to identify characters. However, there are exceptions, such as George Orwell's *Animal Farm*, where most of the characters are animals, yet identifiable as characters due to them having proper nouns as names, e.g. Napoleon and Benjamin. In any instance, such issues are beyond the scope of this thesis.

BERT-Large-NER is occasionally unable to output the entire name as one single entity. In such scenarios, the individual tokens of the name are classified instead. These are referred to as *subwords* by the BERT-Large-NER documentation [31]. The subword is classified as one of the aforementioned class abbreviations prepended by *B-* if it is the beginning of the entity, and *I-* otherwise. For example, BERT-Large-NER recognizes the person name ”Amadeus” as consisting of two subwords, ”Amade” and ”us”, where ”Amade” is classified as B-PER and ”us” as I-PER. If the model succeeds in classifying the entire name as one entity, the entity is classified as B-PER.

Another field in BERT-Large-NER's output is `word`, which is either the subword or, if the model succeeds in recognizing the entire name as an entity, the entire name itself. In this thesis, the word ”word” is used to refer to a word in its original literal sense, and terms such as ”subword”, ”name” or ”the value of `word`” to refer to the field `word` depending on context.

The value of `word` in I-PER entities is prepended with double hashtags (##) if the entity is not preceded by a space. For example, if the subword ”Wolfgang” is classified as ”B-PER” and the subwords ”Amade” and ”us” as I-PER in the name ”Wolfgang Amadeus”, the value of `word` will be ”Amade” for the subword ”Amade” since there is a space between ”Wolfgang” and ”Amade”, and ”##us” for the subword ”us” since there is no space between ”Amade” and ”us”.

As also stated in the BERT-Large-NER documentation, post-processing is inevitable when a name is split into subwords. Moreover, although not explicitly stated in the documentation, the post-processing is most

likely intended to be conducted by removing the double hashtags from the I-PER subwords that contain double hashtags; prepending a space to I-PER subwords that do not contain double hashtags; and concatenating a B-PER entity with all its subsequent subwords. However, subwords that should be I-PER are often misclassified as B-PER and vice versa, making this way of post-processing impossible.

- *Missing subwords*: BERT-Large-NER occassionally fails to classify all subwords of a name, leaving the desired entity fragmented as incohesive subwords. For example, it may be able to classify the subwords ”Pha” and ”oh” but not ”ra” in ”Pharaoh”.

Measures for post-processing To handle the imperfections outlined above, we implemented a four-step post-processing procedure. Before examining the procedure, a few definitions must be introduced:

- $subword_n$ is the *n*:th subword that is to be post-processed.
- $entity_n$ is the *n*:th entity that has been obtained after post-processing a subword.
- $chunk_{subword_n}$ is the chunk from which $subword_n$ has been inferred.
- $word\_index_{entity_n}$ is the index of the *n*th entity relative to all words within $chunk_{subword_n}$ regardless of the words being entities or not, e.g. the index for ”Clara” in the chunk ”My name is Robin and this is Clara.” would be denoted as $index_{entity_2} = 8$ since ”Clara” is preceded by 7 words.
- $last\_letter\_index_{list_n}$ where $list \in \{subword, entity\}$ is the index of the last letter of a subword or entity relative to all letters within $chunk_{subword_n}$ including non-alphabetical characters.

Given the above definitions, the post-processing procedure was implemented as follows:

1. Remove any double hashtags from $subword_n$ if present, which is the case if $subword_n$ is an I-PER not preceded by a space as already elaborated.
2. Concatenate all adjacent alphabetical letters of $subword_n$ within $chunk_{subword_n}$. Specifically, prepend the letter if it precedes $subword_n$, and append it if it is after $subword_n$. Set $entity\_n$ to the result of the concatenation.

3. If $word\_index_{entity_n} - word\_index_{entity_{n-1}} = 1$, i.e. if there is no word between the currently and previously post-processed entities within $chunk_{subword_n}$, append a space followed by $entity_n$ to $entity_{n-1}$, i.e. treat $entity_{n-1}$ and $entity_n$ as the same entity where, for example, the former is the entity's first name and the latter the entity's surname separated by a space. Then, set $last\_letter\_index_{entity_{n-1}}$ to $last\_letter\_index_{entity_n}$.

4. If $last\_letter\_index_{subword_i} <= last\_letter\_index_{entity_{n-1}}$ where $i > n - 1$, then skip the post-processing of that subword. In other words, if $entity_{n-1}$ has already been concatenated with letters belonging to $subword_i$ from step 20, there is no need to post-process $subword_i$.

4.1.1.6 Filtering

The list of entities obtained from the post-processing —referred to as the "post-processing list" —occasionally contained person entities which are not necessarily characters. Moreover, prefixes such as honorifics, titles and salutations, as well as similar character names that the LLM is incapable of distinguishing was also included sometimes in the result of the post-processing. To handle these issue, the following filters were applied to the entities:

- *Remove first name or surname if present in multiple entities*: If a character A and a character B shared the same first name, and the LLM tried to generate a summary for character A from a chunk where B was also present and both characters were referred to by their first name, then the LLM could not distinguish character A from character B. To prevent this issue, the entity which only consisted of the first name shared by character A and character B was removed, while the entities which consist of a combination of character A or character B's first name and surname were kept. For example, the entity `David` would be removed from the list `{David, David Johnson, David Clinton}` after having applied the filter.

  The same filter was applied to the characters' surnames. For example, `Johnson` would be removed from `{Johnson, David Johnson, Sarah Johnson}`.

- *Remove surname in plural form*: Sometimes, families were referred to by the family's surname in plural form, which was recognized by

BERT-Large-NER as an entity. While it is true that a family consists of characters, it was problematic to treat it as an entity on its own since the prompt sent to the LLM expects only one character per summary. To prevent this issue, a list of surnames was first extracted from the entities consisting of full names. Then, any entity that ended on the letter ”s”, and where all letters preceding the ending ”s” constituted a surname that is present in the extracted list of surnames, was removed from the post-processing list.

- *Remove author*: The author of the book designated in the ’creator’ field of the ’DC’ metadata in the ePub file was removed from the post-processing list.

- *Remove prefixes*: Prefixes, such as honorifics, titles and salutations, were removed from the characters’ names so that no variations of the same character were present in the post-processing list. For example, ”Scrooge” and ”Mr. Scrooge” are the same character, yet the summaries produced for them would be treated as summaries for two distinct characters by the hierarchical merging discussed in section 3.2.2. Therefore, ”Mr.” would need to be stripped from ”Mr. Scrooge” in order for the two to be recognized as the same character.

  The prefixes were removed based on pre-defined lists of honorifics, titles, salutations and family relationships such as ”Father”, ”Mother” etc. These lists can be found through the link listed in paragraph A.2.0.0.1 of appendix A.

4.1.2 Summarization

The character summaries were produced in two steps: First, the source text had to be split into appropriately sized chunks for the LLM to process. Then, hierarchical merging as defined in section 3.2.2 was applied to the generated chunks to generate the summaries. These two steps are examined in order below.

4.1.2.1 Chunkification

As discussed in section 1.2.2, and unlike the chunkification process for the character extraction, the summarization was assumed to heavily benefit from having cohesive chunks, i.e. chunks that consist of a discrete number of discourse units. In order to split the source text into cohesive chunks, two

things had to be determined: what discourse unit to use, and how to effectively find the number of discourse units that maximizes the LLM's context.

Discourse unit to use The aim of generating cohesive chunks was to select the source text in such a manner that maximizes the number of interconnected facts to base the summary on, so that the summary becomes as informative possible. An example is provided below to elucidate the benefits of this approach and reflect on what discourse unit that is useful for our purposes:

Assume that the LLM is provided with sentence-level discourse units when summarizing characters in a given book. The book contains the sentence "*As Jasper peered over the cliff, a sense of exhilaration coursed through his veins.*", which is immediately followed by "*He had finally activated his virtual reality headset, stepping into the adventure he'd been anticipating for weeks.*". Then, if the maximum number of sentences that fit within the LLM's context window is selected as the source text, there is a risk that the source text will accommodate the first sentence but not the second, leading to a summary that omits the crucial context provided by the latter sentence.

Segmenting these two sentences into separate chunks could be justified if the author deliberately placed the first sentence at the end of one chapter and the second at the start of another to create a cliffhanger effect. Nonetheless, to ensure such chunkification aligns with the author's intent, chapter-level discourse units would be more appropriate than sentence-level ones. Therefore, the source text should ideally be divided into chapter-level chunks, granted each chapter is small enough to fit within the LLM's context window.

Regrettably, due to inconsistencies in chapter structuring within the ePub files used in this study, a uniform approach for chapter extraction was not feasible. As a result, we opted for sentence-level discourse units when chunkifying the source text.

Discourse unit count that maximizes the context The larger context provided to the LLM, the more facts can be contrasted against each other when determining what facts that are the most relevant for the summary. However, since the context should consist of a discrete number of sentences, and the sentences have varying length, the number of sentences that should constitute each chunk cannot be determined in advance.

The following algorithm was one possible approach to solve the above issue:

1. Append one sentence at a time from the book to the source text.

2. Check if the resulting source text's number of tokens exceeds the context window and act accordingly:

   (a) If the source text exceeds the context window, remove the last appended sentence and stop.

   (b) If the source text does not exceed the context window, repeat the procedures from the beginning.

While this algorithm resolved the issue, it operated with a time complexity of $\mathcal{O}(n)$, both in a best and worst-case scenario, as it appended one sentence at a time. To enhance efficiency, the average number of sentences expected to fit a chunk had to be pre-calculated. Let $sentences_{avg}$ denote this number. Furthermore, let $sentences_{total}$ denote the book's total number of sentences, $tokens_{total}$ its overall token count, and $window$ the maximum number of tokens accepted by the LLM, i.e. the context window. Then, the average tokens per sentence can be expressed as $tokens_{avg} = \frac{tokens_{total}}{sentences_{total}}$. Utilizing this, we could calculate $sentences_{avg}$ as:

$$sentences_{avg} = \frac{window}{tokens_{avg}}$$

To optimize the algorithm's time complexity, we prepended a step before appending one sentence at a time: "Append the first $sentences_{avg}$ number of sentences to the source text". This extra step yielded a best-case time complexity of $\mathcal{O}(1)$ when $sentences_{avg}$ perfectly fitted the context window, and a worst-case complexity of $\mathcal{O}(n)$.

4.1.2.2 Hierarchical Merging

After completing the character extraction discussed in section 4.1.1 and chunkifying the source text from the preceding step, the LLM could commence the hierarchical merging to generate the character summaries. This section elaborates on the prompts based on the format outlined in 3.2.2.2 that were sent to the LLM to execute the summarization and merging, rather than examining the implementation of the algorithm procedures, since it simply adhered to the algorithm described in section 3.2.2.1.

Prompt Templates To maintain consistency in the summaries across characters and books, and to standardize the output format of the LLM such that it is easily processed by the backend service discussed in section 4.2, we utilized prompt templates provided by the LangChain library, introduced

in section 4.1.1. Prompt templates are essentially functions that ensure consistent prompts by returning a string that serves as the base structure of the prompt. These strings are by nature static, but slight modifications are allowed: The prompt template accepts string arguments which are inserted into pre-determined positions within the static string.

Prompt Template #1: Subsummarization Listing B.4 in appendix B illustrates the prompt template used in the first iteration of the hierarchical merging to generate subsummaries.

The prompt template in listing B.4 deserves further explanation in regards to two aspects: its input, and its structure. The prompt template's input consisted of three parameters:

- Some *text* from the book, i.e. the source text.
- The *character names*, structured as a string wherein each name, except the last, is separated by a comma and a space (", "). The final name was separated from the others by the word "and", surrounded by spaces (" and "). An example of such a string would be "Emma, Jones and Scrooge".
- The *character names in JSON format*, e.g. `{ "Emma": "Summary for Emma", "Jones": "Summary for Jones" }`, where the value in the key-value pairs explicitly stated that it is a placeholder for the summary of its associated character, thereby preventing confusion.

The structure of the prompt template can be understood as comprising three main components:

- **Pre-instructions**: These directives instructed the LLM to interpret the prompt as two distinct sections: a set of *instructions* and some *text* extracted from the book. This division ensured that the LLM does not mistakenly treat the book text as part of the instructions.
- **Instructions**: This segment guided the LLM on how to generate its output. It specified that the summaries should be structured as a JSON object with key-value pairs, where the key represented the character and the value denoted the summary content. Additionally, the instructions dictated what aspects of the text the LLM should focus on when creating the summaries, albeit sometimes expressed ambiguously. They may include directives such as: "*For example, you can write about relations*

to other characters, personality traits, appearance, and other things you might find interesting". This was intentional: We noticed that when the LLM was provided with too clear instructions, it explicitly mentioned when it failed to abide by the instructions. For example, the summary sometimes stated that "*[n]o mentions of the character's appearance could be found*", which is not a desirable statement to include in a character summary.

- **Text**: This section contained the actual source text. While a more descriptive title such as "Source Text" or "Excerpt" could have been used, the decision was made to employ the generic term "Text" to prevent the LLM from explicitly referencing the source type when the title was overly specific.

Prompt Template #2: Merging Subsummaries Listing B.5 in appendix B shows the prompt template used for merging the subsummaries to create the final summaries. It took the following arguments as its input:

- One character name.
- One string containing two summaries labeled "Summary #1" and "Summary#2".

Just like the template for creating the subsummaries, this template was also structured as three segments: a set of pre-instructions, instructions and the subsummaries to be merged.

Prompt Examples While prompt template #1 governed the content of the summary, its format was controlled by "prompt examples", which are examples of interaction between the user and the LLM that reinforces the LLM to generate output accordingly. A prompt example consists of two "messages": *user messages*, and *assistant messages*. Moreover, while not strictly being part of the prompt example, these messages are often accompanied by a third message called *system messages*. The role and implementation of each message is detailed below:

- **User Message**: This is essentially the same as a user prompt. In this work, it was identical to prompt template #1 but with the variables replaced with placeholder text. For example, the sentence "*Briefly summarize the characters* ***parsed_character_names*** *based on Text.*"

was replaced by "*Briefly summarize the characters* ***(the characters' names)*** *based on Text.*".

- **Assistant Message**: This message represents the expected output generated by the LLM in response to the user message. In this work, it was written as a JSON object, where each key in its key-value pairs represented a placeholder for the character, and the corresponding value had the following structure: "*(character #1) is (other character's name)'s (relation). She is (personality traits). (description of appearance).*". In other words, the summary was enforced to begin with a description of the character's relations to other characters, followed by their personality traits and lastly their appearance. The LLM tried to abide by these directives if feasible, but was not forced to follow them at all times.

- **System Message**: This message provides the LLM with general instructions on how it should behave. For example, it can set the LLM's tonality in its responses and what it should be as well as not be capable of. In this work, the system message was set to "*You are a book publisher that summarizes a character based on an excerpt from a literary book.*".

Storing Summaries and Sentences Once the summaries had been generated, two CSV files were stored:

- **Summaries**: The final summaries generated after the hierarchical merge were stored in a CSV file containing the following columns:

  - **summary**: The actual summary.
  - **character_name**: The name of the character associated with the summary.
  - **end_sid**: The *sentence ID* of the sentence on which the chunk associated with the summary ends. A sentence ID is not inherent in the ePub file: it is artificially associated with each sentence in the data frame produced by the ePub parser provided by BookBeat, discussed in section 4.1.1.2, and is based on the index of the sentence.

- **Sentences**: All sentences that make up the book were stored in a CSV file containing the following columns:

- **sentence**: The actual sentence.
- **sid**: The sentence ID associated with the sentence.

4.2 Server

The backend consisted of two functionalities: functionality for pre-processing the E-books so that the characters were easily accessible by CSS selectors on the clien, and an API through which the client can access the database (which corresponds to the CSV files discussed under the paragraph ”Storing Summaries” of section 4.1.2.2. This section examines these functionalities.

4.2.1 Character Pre-Processing

To enable users to identify characters with an available summary on a certain page, the aim was to apply a pink highlighting CSS style to these characters. However, to implement this CSS style, it was necessary to modify the HTML content of the E-book itself, as no tools for this purpose were available in the reader library utilized. The approach involved iterating through all characters in a book, and upon detecting a character, enclosing the character in a span with the class `character` used by frontend to identify characters, i.e. `<span class=″character″>character</span>`. To automate the process of inserting these spans, a short Python program was developed that iterates through all books using the ebooklib Python library. After the iteration, the updated E-books were saved locally.

4.2.2 API

The API exposed the database to the client through the following three endpoints:

- **/api/download-epub/{bookName}**: Exposed the E-books as ePub files.
- **/api/sentences/{bookName}**: Exposed a specified book’s sentences and their associated sentence IDs.
- **/api/summaries/{bookName}/{character}/{sid}**: Exposed summaries for a specific character of a specified book, where the summaries are associated with a sentence ID lower than a specified sentence ID.

4.3 Client

In this section, the design of the client is first reviewed before proceeding to the implementation.

4.3.1 Design

The client had been designed to demonstrate a proof of concept of how the character summaries produced by the character summarization service can be consumed within the interface of the E-book. Considerations of BookBeat's existing user interface were incorporated into the design to some extent, but it was by no means intended to be directly transferable to BookBeat's user interface as is. The application was branded as *Character Navigator* to emphasize the application's purpose as a tool for navigating the characters of a book.

Furthermore, the design of the client can be categorized by three labels, which will serve as the subheadings of this section:

- *Miscellaneous*, which refers to views that are irrelevant for the presentation of the application's core functionality, but nonetheless are relevant in providing the user with a holistic experience of the intended usage of the application.
- *Reader*, which is the interface through which the E-book is read.
- *Summary*, which is the interface that displays the character summary.

4.3.1.1 View: Miscellaneous

Two views belong to the Miscellaneous category: the *Login* view in figure 4.1, and the *Book Selector* view in figure 4.2.

The Login view is the first view presented to the user upon opening the application. After having logged in, the user proceeds to the Book Selector view, in which the user selects an E-book to read.

4.3.1.2 View: Reader

Once a book has been selected in the Book Selector view in figure 4.2, the user is presented with the Reader interface in figure 4.3 in which the E-book can be read. In addition to generic functionality present in most E-book readers, the user can highlight the characters for which summaries have been generated by tapping the device screen, as shown in figure 4.4.

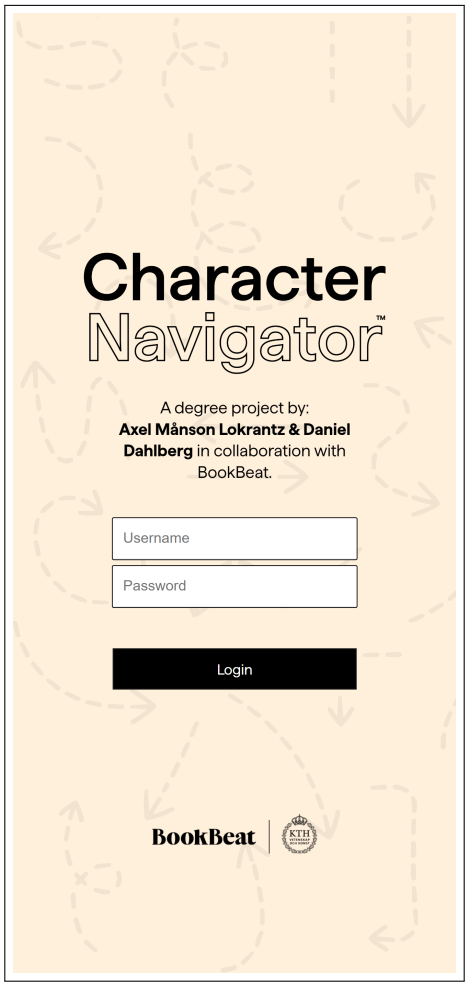


Figure 4.1: Login View

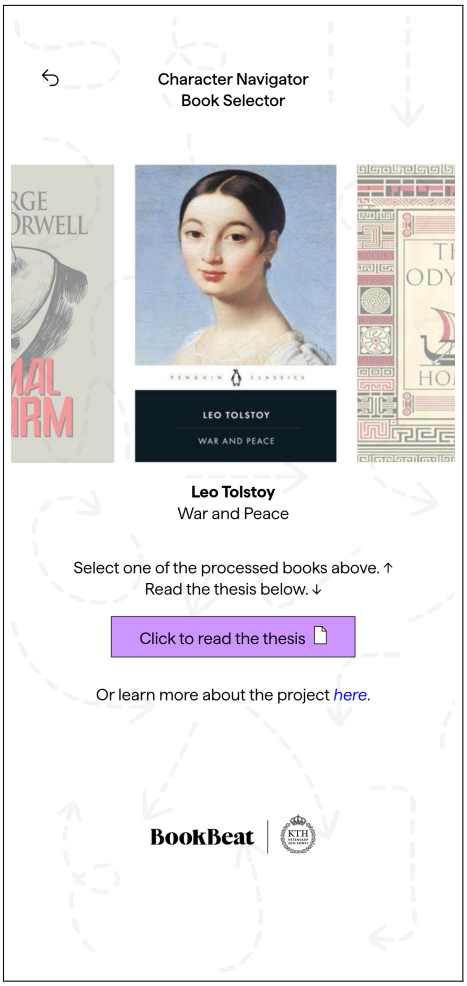
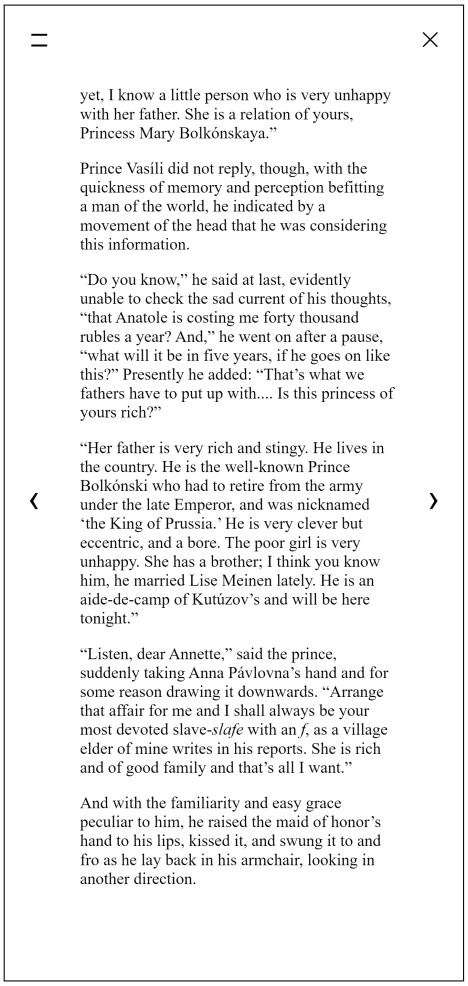


Figure 4.2: Book Selector View



yet, I know a little person who is very unhappy with her father. She is a relation of yours, Princess Mary Bolkónskaya."

Prince Vasíli did not reply, though, with the quickness of memory and perception befitting a man of the world, he indicated by a movement of the head that he was considering this information.

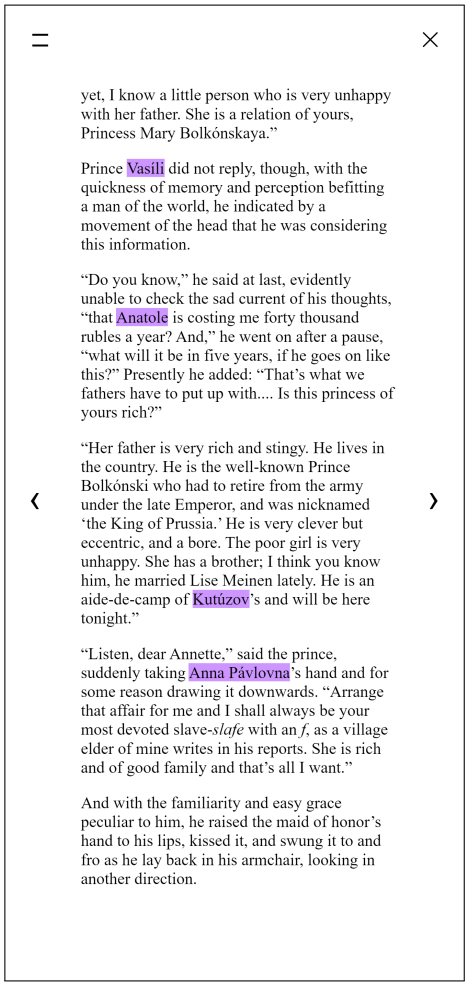
"Do you know," he said at last, evidently unable to check the sad current of his thoughts, "that Anatole is costing me forty thousand rubles a year? And," he went on after a pause, "what will it be in five years, if he goes on like this?" Presently he added: "That's what we fathers have to put up with.... Is this princess of yours rich?"

"Her father is very rich and stingy. He lives in the country. He is the well-known Prince Bolkónski who had to retire from the army under the late Emperor, and was nicknamed 'the King of Prussia.' He is very clever but eccentric, and a bore. The poor girl is very unhappy. She has a brother; I think you know him, he married Lise Meinen lately. He is an aide-de-camp of Kutúzov's and will be here tonight."

"Listen, dear Annette," said the prince, suddenly taking Anna Pávlovna's hand and for some reason drawing it downwards. "Arrange that affair for me and I shall always be your most devoted slave-*slafe* with an *f*, as a village elder of mine writes in his reports. She is rich and of good family and that's all I want."

And with the familiarity and easy grace peculiar to him, he raised the maid of honor's hand to his lips, kissed it, and swung it to and fro as he lay back in his armchair, looking in another direction.

Figure 4.3: Reader View (before tapping)



yet, I know a little person who is very unhappy with her father. She is a relation of yours, Princess Mary Bolkónskaya."

Prince Vasíli did not reply, though, with the quickness of memory and perception befitting a man of the world, he indicated by a movement of the head that he was considering this information.

"Do you know," he said at last, evidently unable to check the sad current of his thoughts, "that Anatole is costing me forty thousand rubles a year? And," he went on after a pause, "what will it be in five years, if he goes on like this?" Presently he added: "That's what we fathers have to put up with.... Is this princess of yours rich?"

"Her father is very rich and stingy. He lives in the country. He is the well-known Prince Bolkónski who had to retire from the army under the late Emperor, and was nicknamed 'the King of Prussia.' He is very clever but eccentric, and a bore. The poor girl is very unhappy. She has a brother; I think you know him, he married Lise Meinen lately. He is an aide-de-camp of Kutúzov's and will be here tonight."

"Listen, dear Annette," said the prince, suddenly taking Anna Pávlovna's hand and for some reason drawing it downwards. "Arrange that affair for me and I shall always be your most devoted slave-*slafe* with an *f*, as a village elder of mine writes in his reports. She is rich and of good family and that's all I want."

And with the familiarity and easy grace peculiar to him, he raised the maid of honor's hand to his lips, kissed it, and swung it to and fro as he lay back in his armchair, looking in another direction.

Figure 4.4: Reader View (after tapping)

4.3.1.3 View: Summary

Upon clicking on a character in the Reader view discussed in section 4.3.1.2, regardless of the character being highlighted or not, the Summary view is made visible as a panel covering the bottom half of the screen as shown in figure 4.5.

The Summary view is divided into two sections:

- *Summary*: At the top of the panel, the summary associated with the position of the page that the user currently reads is displayed.
- *Summary index*: At the bottom of the panel, the summary index, i.e. the extent of the book on which the summary is based, is displayed as:
  - a *progress bar* divided into as many sub-bars as there are chunks for a specific character. The progress bar has two purposes: to show how many chunks that a summary is based on, and to provide an interface for selecting a summary to display. As elaborated in section 3.2.2.1, each summary is based on exactly one chunk. Therefore, a sub-bar in the progress bar is able to represent a chunk and a summary at the same time.

    The progress bar consists of the following components:

    * the filled pink-colored sub-bar represents the currently selected summary, as well as a read chunk.
    * the transparent pink-colored sub-bars represent selectable summaries, as well as read chunks.
    * gray-colored sub-bars represent non-read chunks and are unselectable.
  - a *percentage* representing how much of the book that the summary is based on.

In the event that the user attempts to view the first summary of a character despite not having read the corresponding chunk, the user will be presented with a warning message stating that the summary may contain spoilers. The user can then choose whether to display the summary, or close the warning message. The warning message is shown in figure 4.6.

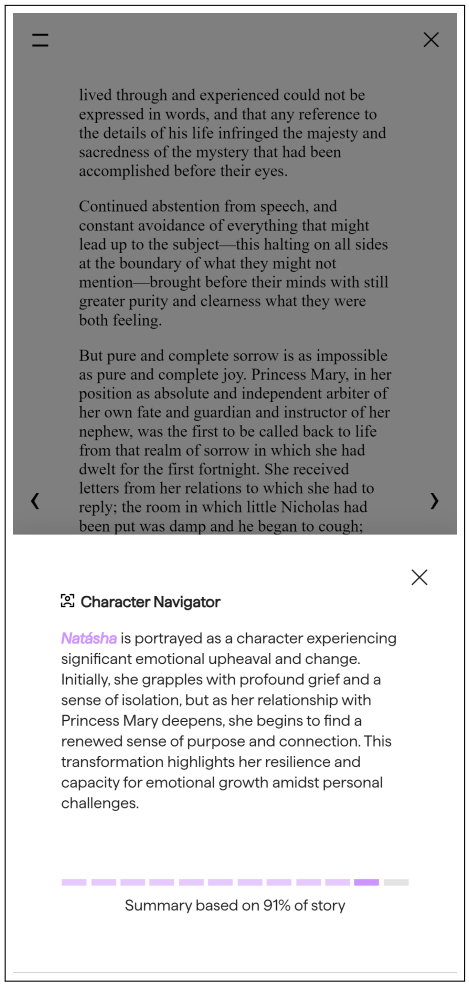


Figure 4.5: Summary View

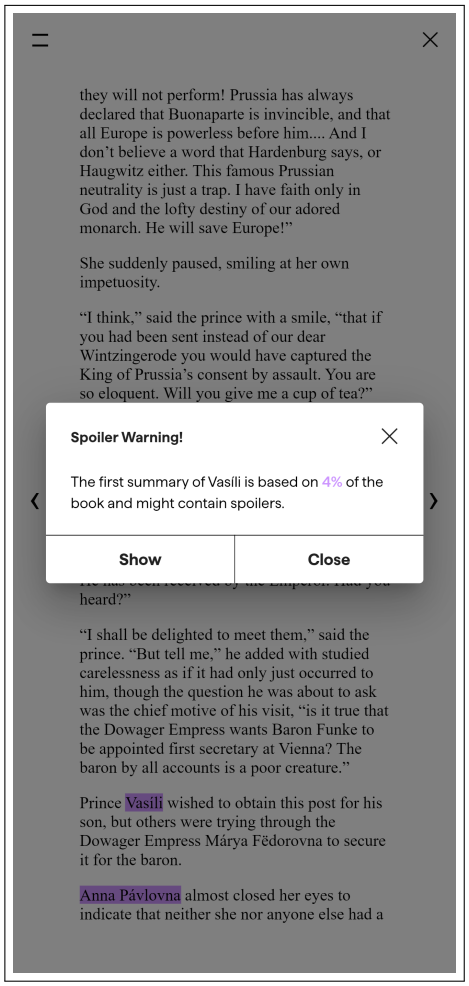


Figure 4.6: Spoiler Warning

4.3.2 Implementation

This section does not go into the details of the implementation of each view discussed in section 4.3.1 as they for the most part consist of straightforward JavaScript, HTML and CSS. The focus instead lies on an issue that was crucial for the fulfillment of one of the objectives that we particularly struggled to solve, namely how to determine what summaries the user should have access to based on their reading progress. The resolution of this issue was broken down into a three-step procedure, elaborated in the following sections.

4.3.2.1 Text Extraction from the Current Page

The client uses an open-source React library named ”react-reader” to display E-books. This library provides a React component, `ReactReader`, which includes a property called `getRendition`. This property accepts a callback function that is invoked each time the `ReactReader` component is re-rendered. The callback receives an argument referred to as the ”rendition object” or simply `rendition`, which contains the current state of the component.

The `rendition` object includes several properties, such as an event listener that captures specific events triggered by user interactions with the E-book. One of these events is `locationChanged`, which occurs when the user navigates to a different page.

Additionally, `rendition` provides information regarding the current page's location within the E-book, identified by Canonical Fragment Identifiers (CFIs). For example, the CFI `epubcfi(/6/14!/4/30/1:87)` points to a specific location in the XHTML structure of the ePub file, with the numbers before the colon representing HTML node indices and the number after the colon indicating a character index within the deepest child node. Although not explicitly stated in any source that we are aware of, the first three nodes appear to denote a higher-level book structure, such as a chapter, while the latter nodes specify the exact location on a page.

To extract the text from the current page, a CFI range, which consists of a parent path and start and end subpaths, had to be constructed using the starting and ending CFIs found in `rendition.location.start` and `rendition.location.end`.

By performing this text extraction process in response to the `locationChanged` event, the client could dynamically retrieve the text content whenever the user navigates to a new page.

4.3.2.2 Determining Sentence ID of the Current Page

The text content retrieved from the current page was split into sentences and then compared one at a time with the entries in the book's sentence list located at the `/api/sentences/{bookName}` endpoint, discussed in section 4.2.2. Upon finding a sentence existing both on the current page and in the sentence list, the ID for that sentence was retrieved.

4.3.2.3 Fetching Summaries Based on Sentence ID

After the sentence ID retrieval described in section 4.3.2.2, the summaries with a sentence ID lower than the retrieved sentence ID were fetched from the endpoint `/api/summaries/{bookName}/{character}/{sid}` described in section 4.2.2.

Chapter 5

Results and Analysis

In this section, the results from evaluating summaries for the characters specified in section 3.4.2 are examined in section 5.1 and then analyzed in sections 5.2 and 5.3. The evaluation was conducted according to the procedures outlined in section 3.7

5.1 Results

The summaries for the characters specified in section 3.4.2 were evaluated through the procedures outlined in section 3.7, yielding the results related to content adherence in tables 5.1 and 5.2, and the results pertaining to format adherence in table 5.3.

Book	Character	F1	Precision	Recall
Animal Farm	Napoleon	0.631	0.639	0.635
	Snowball	0.594	0.534	0.563
	Boxer	0.698	0.591	0.641
The Odyssey	Ulysses	0.628	0.663	0.645
	Telemachus	0.649	0.616	0.632
	Jove	0.725	0.582	0.646
War and Peace	Pierre	0.644	0.631	0.658
	Natásha	0.653	0.716	0.600
	Rostóv	0.634	0.673	0.599
Mean	-	0.65	0.627	0.624
Standard Deviation	-	0.039	0.054	0.03

Table 5.1: BERTScore for the most frequently occuring characters in the sample books.

Book	Character	F1	Precision	Recall
Animal Farm	Minimus	0.665	0.726	0.613
	Pincher	0.543	0.576	0.514
	Jessie	0.632	0.648	0.618
The Odyssey	Helen	0.647	0.649	0.645
	Eurymachus	0.667	0.701	0.635
	Aegisthus	0.613	0.579	0.652
War and Peace	Rostopchín	0.597	0.638	0.561
	Alexander	0.584	0.579	0.590
	Bourienne	0.576	0.620	0.537
Mean	-	0.613	0.635	0.596
Standard Deviation	-	0.042	0.054	0.049

Table 5.2: BERTScore for less frequently occurring characters in the sample books.

Character	$cand_{ordered}$	$cand_{total}$	$ref_{ordered}$	ref_{total}
Napoleon	1	3	1	3
Snowball	1	1	1	3
Boxer	0	2	1	3
Minimus	1	1	3	3
Pincher	2	2	3	3
Jessie	0	3	1	2
Ulysses	1	3	0	3
Telemachus	2	2	3	3
Jove	1	1	3	3
Helen	0	3	3	3
Eurymachus	3	3	3	3
Aegisthus	0	2	3	3
Pierre	0	1	0	3
Natasha	0	1	1	3
Rostóv	0	1	1	3
Rostopchín	0	1	1	3
Alexander	0	1	0	2
Bourienne	2	2	2	2
Mean	0.833	1.777	1.666	2.833
Standard Deviation	0.857	0.808	1.188	0.383

Table 5.3: Scores according to the evaluation framework for format adherence

5.2 Reliability Analysis

The reliability analysis is divided into two sections: one where the reliability of the content adherence is discussed, and one where the reliability of the format adherence is examined.

5.2.1 Reliability Analysis of Content Adherence

Section 3.6.1 defines the reliability of the content adherence related results as measured by computing the confidence interval of the F1 score for the most frequently mentioned characters and the less frequently mentioned characters respectively. The standard deviations for these groups of characters can be seen in tables 5.1 and 5.2. Denote the former group's standard deviation as $s_1 = 0.039$ and the latter as $s_2 = 0.042$, and their corresponding means as $\overline{x}_1 = 0.65$ and $\overline{x}_2 = 0.613$ respectively. Then, the confidence intervals for each respective group of characters can be calculated as follows:

$$CI_1 = \overline{x_1} \pm q^N_{0.95} \times \frac{s_1}{\sqrt{n}} = 0.65 \pm 1.96 \times \frac{0.039}{3} = [0.625, 0.675]$$

$$CI_2 = \overline{x_2} \pm q^N_{0.95} \times \frac{s_2}{\sqrt{n}} = 0.613 \pm 1.96 \times \frac{0.042}{3} = [0.586, 0.64]$$

Since the lower bounds for both intervals are greater than 0.5, the threshold for the BERTScore F1 score that the thesis in section 1.4 set as its aim to surpass, the reliability in terms of content adherence is deemed as sufficiently high, albeit with some caveats:

The summaries for the sampled characters are biased insofar:

- The distribution of frequencies among the less frequently mentioned characters is to some extent uneven, as discussed in section 3.4.2.
- They were sampled from three books that were deliberately selected because their characters having distinct names that are proper nouns, which is not always the case, as noted in passing in section 4.1.1.5. If these criteria are not met, then BERT-Large-NER would most likely not be able to identify some of the characters, leading to biased results. However, since the goal of this thesis is to evaluate the results of hierarchical merging and not character extraction, as stated in section 1.4, such aspects are beyond the scope of this study.

5.2.2 Reliability Analysis of Format Adherence

Section 3.6.2 defines the reliability of format adherence results using the standard deviations of $cand_{ordered}$ and $cand_{total}$. Rather than comparing these values against a threshold, they serve as reference points for a qualitative analysis of format adherence reliability.

First, we analyze the reliability of the order of the candidates' character components. As shown in table 5.3, the standard deviation of $cand_{ordered}$, $s_{cand_{ordered}} = 0.857$, slightly exceeded its mean, $\overline{x}_{cand_{ordered}} = 0.833$. This indicates a high degree of randomness in the order of the character components. Second, although the standard deviation of $cand_{total}$, $s_{cand_{total}} = 0.808$, was smaller than its mean, $\overline{x}_{cand_{total}} = 1.777$, it still suggests that the number of components contained by the candidate frequently fluctuates, on average, between one ($\overline{x}_{cand_{total}} - s_{cand_{total}} \approx 1$) and three ($\overline{x}_{cand_{total}} + s_{cand_{total}} \approx 3$).

In contrast, the reliability of the references was generally higher regarding component inclusion. The standard deviation of ref_{total}, $s_{ref_{total}} = 0.383$, was much lower than its mean, $\overline{x}_{ref_{total}} = 2.8$, indicating that the references rarely omits any components. However, the standard deviation of $ref_{ordered}$, $s_{ref_{ordered}} = 1.188$, was higher than $s_{cand_{ordered}} = 0.857$, suggesting greater randomness in the order of components within the references compared to the candidates. This discrepancy suggests that randomness in the order of components is due to poorly crafted prompt instructions, rather than being a side-effect of hierarchical merging. Nonetheless, the difference between $s_{ref_{total}} = 0.383$ and $s_{cand_{total}} = 0.808$ implies that the candidates may be more apt than the references to inadvertently exclude components during the merging process, highlighting an imperfection of hierarchical merging.

5.3 Validity Analysis

As stated in section 3.6.1, the validity of the content adherence related results was gauged by the mean of the F1 score for the most frequently mentioned characters and the less frequently mentioned characters respectively. Both the mean for the former group of characters, $\overline{x}_1 = 0.65$, and for the latter group, $\overline{x}_2 = 0.613$, are above the validity threshold of 0.6 that the thesis aimed to surpass, as explained in section 1.4.

Chapter 6

Discussion

The reference summaries against which the generated summaries were evaluated should overall be reliable as ground truth, since LLMs have most likely been extensively trained on public domain books and the sample books belong to the public domain. However, this is not guaranteed. Therefore, the references should ideally be crafted by a human who has read the sample books. However, it is important to note that such references available online will likely not follow the same structure as the prompt which was used when generating summaries with hierarchical merging, and thus produce misleading results.

An alternative approach is to evaluate both the generated summary and the reference summary against a human-crafted reference to measure their respective fidelity to the source. If the reference summary achieves a lower BERTScore than the generated summary, it should be invalidated. In other words, for the reference summary to be considered valid, its BERTScore should always be higher than that of the generated summary, indicating greater faithfulness to the source. While this approach could have been feasible, it was conceived in the later stages of the thesis, leaving insufficient time to explore it and find such human-made references.

Another aspect to consider when analyzing the results is the extent of GPT-4's background knowledge of the selected books. One way of ascertaining that such background knowledge did not affect the results is by evaluating hierarchical merging when used on recent fan fiction that is not part of GPT-4's knowledge. However, since the references cannot be produced from GPT-4's innate knowledge in the case of fan fiction, the references must either be retrieved externally or crafted manually. Unfortunately, neither option was feasible for this thesis: Character summaries for fan fiction are difficult to

find, especially ones that match the proposed summary structure, and creating them manually requires time and literary expertise beyond the scope of this work.

Chapter 7

Conclusions and Future work

7.1 Conclusions

This thesis explored the automatic generation of character summaries using machine learning, presented to users through an E-book reader. Although all planned functionalities were developed, only the machine learning component for summarizing characters was evaluated.

The first goal in section 1.4 was not met: the generated character summaries did not adhere to the prescribed format. The specified order of character components was not followed, and some components were occasionally excluded. However, the second goal was achieved: the BERTScore thresholds for reliability and validity were surpassed.

Improving format adherence may require transdisciplinary efforts, as the format used was based on our personal preferences, with our limited experience in literary summarization. This was evident in the generated summaries: the prompt allowed the LLM to include additional elements it found interesting. We expected this would occasionally add small amounts of complementary text, but it often resulted in more text than all the specified components together, indicating the format was suboptimal.

While the content adherence thresholds were met, the results were not impressive and have room for improvement. This could be addressed by fine-tuning the machine learning models to the study's specific needs or by employing new models, which are rapidly released in this era of AI. Towards the end of our work, a new version of GPT-4, GPT-4-turbo, was released, increasing the input tokens from 8192 to 128 000. This allows for more cost-efficient summarization and new possibilities. For instance, inferring narratological meta concepts, such as a character's role, is challenging without

processing large inputs of source text, which was not possible with an 8192-token context window. Therefore, some results of this thesis may soon be outdated. However, we hope the thesis provides a general guideline for generating and presenting spoiler-free character summaries in a user-friendly manner.

7.2 Limitations

This section is dedicated to discussion about limitations in regards to two main functionalities: the character extraction, and the summarization.

7.2.1 Character Extraction

Three limitations can be identified for the character extraction:

- **Pre-Processing**: The paratext was removed from the ePub file to minimize the risk of including people that are not characters in the character extraction. To do this, any section in the book which heading was included in the list of paratext headings in listing 4.1.1.1 was eliminated. While this approach is straightforward and effective in most cases, it might inadvertently remove sections that are part of the story if their heading consists of any word that is present in the list of paratext headings.
- **Recognizing Non-Human Characters**: The Named Entity Recognition model BERT-Large-NER was used to identify the characters within a book by letting it determine whether an entity is a person or not. However, as touched upon in section 4.1.1.5, characters can also be animals, inanimate objects, or other entities.

7.2.2 Summarization

The summarization has two notable limitations:

- **Avoid Mentions of Absence**: The prompt used to summarize the characters occasionally produced outputs where the absence of character traits that the LLM was told to focus on is mentioned, despite the LLM being instructed to exclude such mentions as they are undesirable in a summary. We later came to discover that LLMs inherently struggle with negative instructions. This limitation was not apparent to us

until the final stages of our work, despite being documented in prompt engineering texts [32]. This oversight indicates shortcomings in our literature review.

- **Further Analysis of Ideal Summary Structure**: The structure for character summaries proposed in this thesis is based on our subjective opinion and requires further analysis to determine the ideal structure.

7.3 Future work

This section focuses on the limitations of our work listed in section 7.2 and suggests potential solutions, as outlined below.

- **Alternative Method for Pre-Processing**: The pre-processing of the ePub file is relatively simple and straightforward. For the small set of books used in this work, the solution works as intended. However, if the work were to reach a larger scale, a different approach would be more appropriate. ePub files vary significantly in their structure and behaviour, so there is no one-size-fits-all solution. An alternative would be to use an LLM to scan the book for paratext sections and make more nuanced decisions about their removal.

- **Improved Named Entity Recognition**: One of the biggest challenges of this work was the extraction of characters in the ePub books. We used several strategies as outlined under the ”Post-processing and Filtering” bullet point of the list detailing the character extraction procedures in section 3.2.1. BERT-Large-NER has been trained on the CoNLL-2003 Named Entity Recognition dataset to recognize the types of entities mentioned in section 4.1.1.5. However, this work only utilizes the *PER* tag. A reasonable approach for improvement would be to use the BERT-Large-NER as a foundation and then further train it on a more niche dataset that focuses solely on recognizing character names to achieve more accurate outputs.

- **Experimentation with Summarization**: As described in section 7.2.2, the LLM struggles with negative instructions. Even though the prompt clearly states what the LLM should include and exclude in a summary, the LLM seldom excludes what it should exclude. However, there are ways to potentially counter this issue, one being to further experiment with different prompts to see how it reacts. While some experimentation

was done during the work, it could be further explored. Another approach would be to adjust the temperature and top-p values of the model. Temperature and top-p are defined below, based on the definitions provided in [6]:

- **Temperature**: Controls how random the output will be. The lower the temperature the more conservative the model behaves, resulting in more predictable output, while a higher temperature increases creativity.
- **Top-p**: Decides the number of possible words the LLM should consider. It ranges from 0 to 1, where a high value means the model looks at more possible words, and a lower value means it looks at fewer.

Making small adjustments to the prompt, in combination with tweaking the temperature and top-p values, could yield more accurate results. It should also be mentioned that, as we used GPT-4, future models such as GPT-5 could potentially be more accurate, and this problem might not exist.

7.3.1 What has been left undone?

Upon reviewing the section ”Within Scope” 1.6.1, we can conclude that all key points have been fulfilled. As for the section ”Potentially within Scope” 1.6.2, none of the key points have been accomplished.

- Crafting a feature for summarizing entities within the text other than characters, such as locations or events.
  - While the idea is interesting and could be a valuable addition to the prototype, we chose to focus on refining the character extraction process to achieve the best possible results. Nonetheless, since the named entity recognition model can also identify locations, experimenting with this feature remains a feasible option that wouldn't require excessive time.
- Generating general summaries of E-books that capture the narrative up to a designated point in the plot.
  - This is another interesting idea that would not require too much work, as the foundation for this feature is already in place. However, due to time constraints, it was put aside.

- Developing an interface that allows users to access character summaries while listening to audio books.
  - After meetings with our supervisors at BookBeat, we learned that functionality indirectly eliminating the need for such an interface was already implemented in the BookBeat application. Therefore, we decided to prioritize other, more important tasks.
- Refining the character summary generation process by employing more sophisticated evaluation metrics.
  - In section 2.5, the most widely adopted evaluation metrics for text summarizing were reviewed. However, more advanced metrics tailored to specific purposes also exist, which could potentially be interesting for evaluating the results. Nonetheless, after meetings with our supervisor, we decided to stick with BERTScore, as it was straightforward, easy to understand, and relevant for the purpose of the project.

7.4 Reflections

This thesis has the potential to contribute to SDG number 4, ”Ensure inclusive and equitable quality education and promote lifelong learning opportunities for all” [33], by helping students with reading comprehension difficulties more easily navigate characters in assigned class readings. However, evaluating the extent of this contribution to reading comprehension is beyond the scope of this thesis and remains to be explored.

Additionally, the thesis may also support SDG number 8, ”Promote sustained, inclusive and sustainable economic growth, full and productive employment, and decent work for all” [34], by enhancing the reading experience of E-books. This improvement could, in turn, stimulate the consumption of E-books.

References

[1] J. Andersson, "Svenskarna och internet 2023," 2023. [Online]. Available: https://svenskarnaochinternet.se/rapporter/svenskarna-och-internet-2023/bocker-och-poddar/ [Pages 1 and 5.]

[2] A. Mangen, G. Olivier, and J.-L. Velay, "Comparing Comprehension of a Long Text Read in Print Book and on Kindle: Where in the Text and When in the Story?" *Frontiers in Psychology*, vol. 10, p. 38, Feb. 2019. doi: 10.3389/fpsyg.2019.00038. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6384527/ [Pages 1 and 2.]

[3] L.-Y. Li, G.-D. Chen, and S.-J. Yang, "Construction of cognitive maps to improve e-book reading and navigation," *Computers & Education*, vol. 60, no. 1, pp. 32–39, Jan. 2013. doi: 10.1016/j.compedu.2012.07.010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360131512001704 [Pages 1 and 2.]

[4] "Prompt Engineering Guide." [Online]. Available: https://www.promptingguide.ai/ [Page 4.]

[5] S. Peckham, "Getting started with LLM prompt engineering," Mar. 2024. [Online]. Available: https://learn.microsoft.com/en-us/ai/playbook/technology-guidance/generative-ai/working-with-llms/prompt-engineering [Pages 4 and 12.]

[6] "Mastering Top P and Temperature Parameters in Language Models," Feb. 2024. [Online]. Available: https://www.toolify.ai/ai-news/mastering-top-p-and-temperature-parameters-in-language-models-1702187 [Pages 4 and 68.]

[7] "Papers – Nextra." [Online]. Available: https://www.promptingguide.ai/papers [Page 4.]

[8] Y. Chang, K. Lo, T. Goyal, and M. Iyyer, "BooookScore: A systematic exploration of book-length summarization in the era of LLMs," Oct. 2023, arXiv:2310.00785 [cs]. [Online]. Available: http://arxiv.org/abs/2310.00785 [Pages 4 and 16.]

[9] M. Axelsson, "PISA: internationell studie om 15-åringars kunskaper i matematik, naturvetenskap och läsförståelse," 2023. [Online]. Available: https://www.skolverket.se/skolutveckling/forskning-och-utvarderingar/internationella-jamforande-studier-pa-utbildningsomradet/pisa-internationell-studie-om-15-aringars-kunskaper-i-matematik-naturvetenskap-och-lasforstaelse [Page 5.]

[10] "urchade/gliner_multi-v2.1 · Hugging Face," Mar. 2024. [Online]. Available: https://huggingface.co/urchade/gliner_multi-v2.1 [Page 7.]

[11] "What Are Large Language Models (LLMs)? | IBM," Nov. 2023. [Online]. Available: https://www.ibm.com/topics/large-language-models [Page 11.]

[12] "linguistic unit." [Online]. Available: https://www.thefreedictionary.com/linguistic+unit [Page 12.]

[13] J. MSV, "The Building Blocks of LLMs: Vectors, Tokens and Embeddings," Feb. 2024. [Online]. Available: https://thenewstack.io/the-building-blocks-of-llms-vectors-tokens-and-embeddings/ [Page 12.]

[14] L. Degand and A. C. Simon, "On identifying basic discourse units in speech: theoretical and empirical issues," *Discours. Revue de linguistique, psycholinguistique et informatique. A journal of linguistics, psycholinguistics and computational linguistics*, no. 4, Jun. 2009. doi: 10.4000/discours.5852 Number: 4 Publisher: Presses universitaires de Caen. [Online]. Available: https://journals.openedition.org/discours/5852 [Page 12.]

[15] "Understanding Large Language Models Context Windows | Appen." [Online]. Available: https://www.appen.com/blog/understanding-large-language-models-context-windows [Page 12.]

[16] W. Zhang, *Automatic summarization of fiction by generating character descriptions*. McGill University (Canada), 2017. [Online]. Available: https://search.proquest.com/openview/ee07e7844688a8c1b3a766776cfc9e64/1?pq-origsite=gscholar&cbl=18750&diss=y [Pages 13 and 14.]

[17] W. Zhang, J. C. K. Cheung, and J. Oren, "Generating Character Descriptions for Automatic Summarization of Fiction," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 7476–7483, Jul. 2019. doi: 10.1609/aaai.v33i01.33017476 Number: 01. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/4738 [Page 14.]

[18] K. Hines, "History Of ChatGPT: A Timeline Of The Meteoric Rise Of Generative AI Chatbots," Jun. 2023. [Online]. Available: https://www.searchenginejournal.com/history-of-chatgpt-timeline/488370/ [Page 16.]

[19] E. Yan, "Evaluation & Hallucination Detection for Abstractive Summaries," Sep. 2023, section: posts. [Online]. Available: https://eugeneyan.com/writing/abstractive/ [Pages 17 and 20.]

[20] J. Han, M. Kamber, and J. Pei, "2 - Getting to Know Your Data," in *Data Mining (Third Edition)*, ser. The Morgan Kaufmann Series in Data Management Systems, J. Han, M. Kamber, and J. Pei, Eds. Boston: Morgan Kaufmann, Jan. 2012, pp. 39–82. ISBN 978-0-12-381479-1. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780123814791000022 [Page 18.]

[21] D. L. Olson and D. Delen, "Performance Evaluation for Predictive Modeling," in *Advanced Data Mining Techniques*, D. L. Olson and D. Delen, Eds. Berlin, Heidelberg: Springer, 2008, pp. 137–147. ISBN 978-3-540-76917-0. [Online]. Available: https://doi.org/10.1007/978-3-540-76917-0_9 [Page 18.]

[22] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "BERTScore: Evaluating Text Generation with BERT," Feb. 2020, arXiv:1904.09675 [cs]. [Online]. Available: http://arxiv.org/abs/1904.09675 [Page 19.]

[23] "Analyzing machine learning model performance | IBM Cloud Docs." [Online]. Available: https://cloud.ibm.com/docs/watson-knowledge-studio?topic=watson-knowledge-studio-evaluate-ml#evaluate-mllowf1 [Page 21.]

[24] "What is a good F1 score? Simply explained (2022)," Apr. 2022. [Online]. Available: https://stephenallwright.com/good-f1-score/ [Page 21.]

[25] "F1 Score in Machine Learning." [Online]. Available: https://serokell.io/blog/a-guide-to-f1-score [Page 21.]

[26] G. Ryan, "Introduction to positivism, interpretivism and critical theory," *Nurse Researcher*, vol. 25, no. 4, pp. 14–20, Mar. 2018. doi: 10.7748/nr.2018.e1466 [Page 29.]

[27] "What are tokens and how to count them? | OpenAI Help Center." [Online]. Available: https://help.openai.com/en/articles/4936856-what-are-tokens-and-how-to-count-them [Page 30.]

[28] L. Tolstoy, *War and Peace*, reprint edition ed. Penguin Books, Feb. 2009. ISBN 978-0-14-044793-4 [Page 30.]

[29] F. Middleton, "Reliability vs. Validity in Research | Difference, Types and Examples," Jul. 2019. [Online]. Available: https://www.scribbr.com/methodology/reliability-vs-validity/ [Page 33.]

[30] "Evaluating GPT-4 Zero-Shot Summarization for Streamlining Workflows | Width.ai." [Online]. Available: https://www.width.ai/post/evaluating-gpt-4-zero-shot-summarization [Page 35.]

[31] "dslim/bert-large-NER · Hugging Face," Jan. 2024. [Online]. Available: https://huggingface.co/dslim/bert-large-NER [Page 40.]

[32] "Prompt Engineering Guide." [Online]. Available: https://www.promptingguide.ai/introduction/tips#to-do-or-not-to-do [Page 67.]

[33] "Goal 4 | Department of Economic and Social Affairs." [Online]. Available: https://sdgs.un.org/goals/goal4 [Page 69.]

[34] "Goal 8 | Department of Economic and Social Affairs." [Online]. Available: https://sdgs.un.org/goals/goal8 [Page 69.]

Appendix A

Links

A.1 Character Extraction Lists

A.1.0.0.1 Animal Farm https://github.com/grad-thesis/ml-service/blob/master/data/characters/animal-farm_characters.csv

A.1.0.0.2 The Odyssey https://github.com/grad-thesis/ml-service/blob/master/data/characters/the-odyssey_characters.csv

A.1.0.0.3 War and Peace https://github.com/grad-thesis/ml-service/blob/master/data/characters/war-and-peace_characters.csv

A.2 Other

A.2.0.0.1 Filtering Lists https://github.com/grad-thesis/ml-service/tree/master/data/filter-lists

A.2.0.0.2 Requirements https://github.com/grad-thesis/ml-service/blob/master/requirements.txt

A.2.0.0.3 Sample Summaries https://github.com/grad-thesis/ml-service/blob/master/data/evaluation/sample_summaries.xlsx

A.2.0.0.4 ePub parser https://github.com/grad-thesis/ml-service/blob/master/util/ebook_parser.py

Appendix B

Code

B.1 Miscellaneous

Listing B.1: Words frequently recurring in paratext headings

```
["title", "copyright", "dedication", "epigraph",
"table␣of␣contents", "contents", "foreword",
"preface", "acknowledgements", "introduction",
"prologue", "epilogue", "afterword", "footnote",
"endnote", "glossary", "index", "appendix",
"appendices", "illustration", "bibliography",
"references", "blurb", "praise", "biography",
"project␣gutenberg"]
```

Listing B.2: Function for chunkifying the source text

```
def chunkify(tokenizer, sentences_df):
    merged_sentences = '␣'.join(
        sentences_df['sentence']
    ) # Merges the individual sentences of the book
      # into one string containing the whole book
    text_splitter = RecursiveCharacterTextSplitter
        .from_huggingface_tokenizer(
            tokenizer,
            chunk_size = 512,
            chunk_overlap = 0
        )
    ner_chunks = text_splitter.split_text(
```

```
        merged_sentences
    )

    return ner_chunks
```

Listing B.3: Testing

```
def extract_character_names(ner_chunks, tokenizer):
    nlp = pipeline(
        "ner",
        model=bert_model,
        tokenizer=tokenizer
    )
    final_names = []

    for chunk_index, ner_chunk
        in enumerate(ner_chunks):
        names = []
        ner_results = nlp(ner_chunk)

        ...Post-processing...

    return final_names
```

B.2 Prompts

Listing B.4: Prompt template for generating subsummaries

```
summarize_character_from_excerpt_template =
    PromptTemplate(
        input_variables=[
            "text",
            "parsed_character_names",
            "parsed_character_json_format"
        ],
        template="""
        I will provide you with a set of instructions,
        followed by some text. The former is labeled
        'Instructions' and the latter 'Text'.
```

```
        Instructions:
        Briefly summarize the characters
        {parsed_character_names} based on Text.
        For example, you can write about relations to
        other characters, personality traits, appearance,
        or other things you might find interesting.
        Do not mention the source, i.e. that it is
        a text, book or summary anywhere.
        Return the summaries as a JSON object according
        to the following format:

        {{
            {parsed_character_json_format}
        }}

        Text:
        {text}"""
    )
```

Listing B.5: Prompt template for merging subsummaries

```
summarize_character_from_summaries_template =
    PromptTemplate(
        input_variables=[
            "character_name",
            "summaries"
        ],
        template="""
        I will provide you with a set of instructions,
        followed by a sequence of summaries.
        The summaries are labeled 'Summary #1' and 'Summary #2'.

        Instructions:
        Briefly summarize the character {character_name}
        based on Summary #1 and Summary #2.

        {summaries}
        """
    )
```

Appendix C

Planned Implementation of Hierarchical Merging

1. The E-book underwent segmentation into cohesive chunks.
2. The first summary of a given character was based on the initial chunk.
3. **Iteration 1**: Subsequent summaries for chunk C_n were iteratively refined based on the previous summary S_{n-1} and the current chunk.

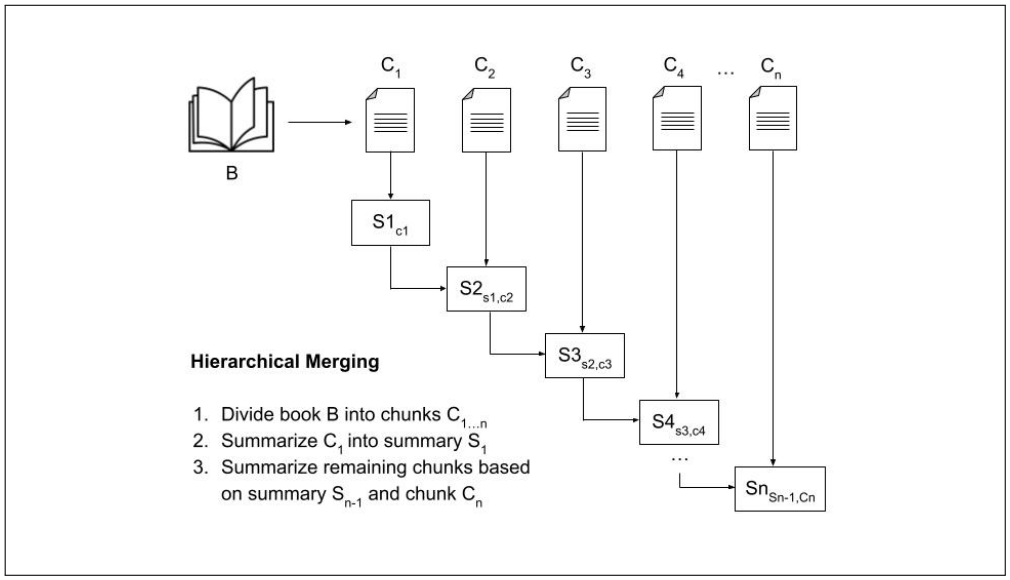


Figure C.1: Hierarchical Merging

TRITA-EECS-EX- 2023:0000

€€€€ For DIVA €€€€

{
”Author1”: { ”Last name”: ”Dahlberg”,
”First name”: ”Daniel”,
”Local User Id”: ”u1zyzalt”,
”E-mail”: ”ddahlb@kth.se”,
”organisation”: {”L1”: ”School of Electrical Engineering and Computer Science”,
}
},
”Author2”: { ”Last name”: ”Månson Lokrantz”,
”First name”: ”Axel”,
”Local User Id”: ”u100002”,
”E-mail”: ”axelml@kth.se”,
”organisation”: {”L1”: ”School of Architecture and the Built Environment”,
}
},
”Cycle”: ”1”,
”Course code”: ”IA150X”,
”Credits”: ”15.0”,
”Degree1”: {”Educational program”: ”Degree Programme in Computer Engineering”
,”programcode”: ”TIDAB”
,”Degree”: ”Bachelors degree”
,”subjectArea”: ”Computer Engineering”
},
”Title”: {
”Main title”: ”Character Navigator”,
”Subtitle”: ”Automated Summarization of Characters in E-Books”,
”Language”: ”eng” },
”Alternative title”: {
”Main title”: ”Character Navigator”,
”Subtitle”: ”Automatiserad summering av karaktärer i E-böcker”,
”Language”: ”swe”
},
”Supervisor1”: { ”Last name”: ”Gauraha”,
”First name”: ”Niharika”,
”Local User Id”: ”u10sbfc6”,
”E-mail”: ”niharika@kth.se”,
”organisation”: {”L1”: ”School of Electrical Engineering and Computer Science”,
”L2”: ”Computer Science” }
},
”Supervisor2”: { ”Last name”: ”Erlansson”,
”First name”: ”Nils”,
”E-mail”: ”nils.erlanson@bookbeat.com”,
”Other organisation”: ”BookBeat”
},
”Supervisor3”: { ”Last name”: ”Arbin”,
”First name”: ”Niklas”,
”E-mail”: ”niklas.arbin@bookbeat.com”,
”Other organisation”: ”BookBeat”
},
”Examiner1”: { ”Last name”: ”Kilander”,
”First name”: ”Fredrik”,
”Local User Id”: ”u1kv5y5m”,
”E-mail”: ”fki@kth.se”,
”organisation”: {”L1”: ””,
”L2”: ”Communication Systems” }
},
”Cooperation”: { ”Partner_name”: ”BookBeat”},
"National Subject Categories": "10201, 10206",
”Other information”: {”Year”: ”2024”, ”Number of pages”: ”1,80”},
”Copyrightleft”: ”copyright”,
”Series”: { ”Title of series”: ”TRITA-EECS-EX” , ”No. in series”: ”2023:0000” },
”Opponents”: { ”Name”: ”A. B. Normal & A. X. E. Normalè”},
”Presentation”: { ”Date”: ”2022-03-15 13:00”
,”Language”:”eng”
,”Room”: ”via Zoom https://kth-se.zoom.us/j/ddddddddddd”
,”Address”: ”Isafjordsgatan 22 (Kistagången 16)”
,”City”: ”Stockholm” },
”Number of lang instances”: ”2”,
”Abstract[eng]”: €€€€
€€€€,
”Keywords[eng]”: €€€€
E-books, Character Summaries, Machine Learning, Large Language Models, Named Entity Recognition, ML, LLM, NER €€€€,
”Abstract[swe]”: €€€€
€€€€,
”Keywords[swe]”: €€€€
E-böcker, Karaktärssammanfattningar, Maskininlärning, Stora språkmodeller, Namngiven entitetsigenkänning, ML, LLM, NER €€€€,

}

acronyms.tex

```
%%% Local Variables:
%%% mode: latex
%%% TeX-master: t
%%% End:
% The following command is used with glossaries-extra
\setabbreviationstyle[acronym]{long-short}
% The form of the entries in this file is \newacronym{label}{acronym}{phrase}
%                                     or \newacronym[options]{label}{acronym}{phrase}
% see "User Manual for glossaries.sty" for the  details about the options, one example is shown below
% note the specification of the long form plural in the line below
\newacronym[longplural={Debugging Information Entities}]{DIE}{DIE}{Debugging Information Entity}
%
% The following example also uses options
\newacronym[shortplural={OSes}, firstplural={operating systems (OSes)}]{OS}{OS}{operating system}

% note the use of a non-breaking dash in long text for the following acronym
\newacronym{IQL}{IQL}{Independent -QLearning}

% example of putting in a trademark on first expansion
\newacronym[first={NVIDIA OpenSHMEM Library (NVSHMEM\texttrademark)}]{NVSHMEM}{NVSHMEM}{NVIDIA OpenSHMEM Library}

\newacronym{KTH}{KTH}{KTH Royal Institute of Technology}

\newacronym{LAN}{LAN}{Local Area Network}
\newacronym{VM}{VM}{virtual machine}
% note the use of a non-breaking dash in the following acronym
\newacronym{WiFi}{-WiFi}{Wireless Fidelity}

\newacronym{WLAN}{WLAN}{Wireless Local Area Network}
\newacronym{UN}{UN}{United Nations}
\newacronym{SDG}{SDG}{Sustainable Development Goal}

\newacronym{LLM}{LLM}{Large Language Model}
\newacronym{NER}{NER}{Named Entity Recognition}
\newacronym{ROUGE}{ROUGE}{Recall-Oriented Understudy for Gisting Evaluation}
\newacronym{DOD}{DOD}{Definition of Done}
\newacronym{ePub}{ePub}{Electronic Publication}
\newacronym{CFI}{CFI}{Canonical Fragment Identifier}
```