



Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Masterthesis

Retrieval Augmented Generation of Character Profiles

Daniel Djahangir

daniel.djahangir@uni-hamburg.de

Studiengang Informatik

Matr.-Nr. 6803168

Erstgutachter: Hans Ole Hatzel

Zweitgutachter: Prof. Dr. Chris Biemann

Abgabe: 14.08.2024

Contents

1	Introduction	1
2	Background	3
2.1	Tokenization	3
2.2	The Transformer	4
2.2.1	Encoder	5
2.2.2	Decoder	5
2.3	BERT	6
2.3.1	Embeddings	6
2.4	Fine-Tuning	7
2.5	Quantization	7
2.5.1	BERTScore	8
2.6	BLEU-Score	9
2.7	RAG	10
3	Related Work	11
3.1	Coreference in Long Documents using Hierarchical Entity Merging	11
3.2	Evaluating Character Understanding of Large Language Models via Character Profiling from Fictional Works	12
3.3	Project Gutenberg	12
3.4	LISCU	13
3.5	Dated Data: Tracing Knowledge Cutoffs in Large Language Models	13
3.6	Methodological Approach	14
4	Gathering of Literature	17
4.1	Querying Wikidata	19
4.1.1	Scraping Fandom Articles	20
5	Experiments	23
5.1	Baseline Experiment	23
5.1.1	Results	26
5.1.2	Analysis	31
5.2	Passage Embedding Experiment	33
5.2.1	Results	34
5.2.2	Analysis	35

5.3	Method Comparison	38
5.3.1	Results	38
5.3.2	Analysis	40
6	Conclusion	45
6.1	Summary	45
6.2	Future Work	45
6.3	Acknowledgments	47
7	Appendix	49
	Bibliography	51
	Affidavit	55

1 Introduction

Embarking on the literary journey of a captivating book, one often finds oneself entangled in a web of intricate characters, each contributing to the rich tapestry of the narrative. Yet, the overwhelming abundance of details can pose a challenge, making it difficult for readers to retain a comprehensive understanding of each character's essence. To address this challenge, this thesis aims to employ Retrieval-Augmented Generation (RAG) in order to create detailed personal descriptions for each character.

An important aspect of this study involves compiling existing literature and human-written descriptions of its characters, allowing for a comparison between these established descriptions and the results generated by large language models (LLMs). We tested various prompts created using different techniques that extract information from the relevant work of fiction. The main focus will be on methods for extracting and modifying information from literature to enhance the query results, along with their evaluation and analysis.

The findings of this thesis can be significantly applied to similar Natural Language Processing (NLP) tasks, especially those requiring the extraction and summarization of information from very long texts. Ultimately, it can also be used to improve the readers' comprehension and as an educational tool that can be used to specifically extract information about a certain entities from texts. It is particularly interesting to observe how LLMs respond to slight variations in query formulations and how well RAG can enhance the results.

Following this brief introduction, I will present a comprehensive overview of the background, focusing on key NLP methods. I will then summarize significant works and papers relevant to my topic before conducting experiments, which I will discuss and evaluate individually. Finally, I will address potential issues encountered during the experiments and suggest possible improvements for the future.

2 Background

The background section explains basic NLP methods in order to gain important knowledge on which I can build as we continue to explore.

2.1 Tokenization

Tokens are the fundamental units of data processing in natural language processing (NLP). A token is the smallest meaningful unit of text, which can be a word, subword, or even a single character or punctuation mark. Tokenization is typically performed at one of three levels: single characters (character-based tokenization), subwords (subword-based tokenization), or whole words (word-based tokenization).

In most modern NLP models, subword tokenization is predominantly used. This technique breaks words into smaller units, such as prefixes and suffixes. Unlike word-based tokenizers, which generate a very large vocabulary and suffer from a loss of meaning across very similar words as well as a large quantity of out-of-vocabulary tokens, or character-based tokenization, where each token has minimal meaning in context and the overall number of tokens on a tokenized text is enormous, subword-based tokenization seeks to find a middle ground. The idea is to decompose rare words into meaningful subwords while maintaining few to single tokens for every meaningful or frequently used word.

Byte Pair Encoding (BPE) is a simple and popular method for subword tokenization. BPE describes an algorithm of frequency-based merging of character pairs into subwords. It starts out with single character symbols and iteratively creates new candidates by merging the most frequent pairs of adjacent symbols until a specified vocabulary size is reached. Another popular tokenizer is the WordPiece tokenizer. The WordPiece tokenizer is quite similar, but uses a different merging criteria. Instead of choosing the most frequent pairs, it chooses pairs that maximize the likelihood of the training corpus by dividing the frequency of the pair by the product of frequency of the individual tokens.

Now both of these methods require a pre-tokenized text or whitespace boundaries. However, not all languages use spaces to separate words such as in Chinese or Japanese. A more general approach than using specific pre-tokenizers is SentencePiece tokenization, that treats the input as a raw input stream, thus including the space in the set of characters to use. It then uses BPE or unigram tokenization, a probabilistic method that selects the best segmentation of text based on the likelihood of subwords to make a purely end-to-end and language independent system.

Subword tokenizers are employed in almost every widely used large language model such as GPT-2, Llama 3, and in large pre-trained language models such as BERT.[Fac24]

2.2 The Transformer

The Transformer architecture, introduced in June 2017 [Vas+17], marked a significant advancement in natural language processing (NLP), initially focusing on sequence-to-sequence NLP problems like machine translation tasks. However, its capabilities quickly revealed a broader potential, particularly in developing large language models (LLMs). These models are trained on vast amounts of raw text using self-supervised learning, a method where the training objective is derived automatically from the input data, typically through tasks such as masked language modeling or causal language modeling. These techniques help the model develop a statistical understanding of the language. The Transformer architecture consists of an encoder and a decoder.

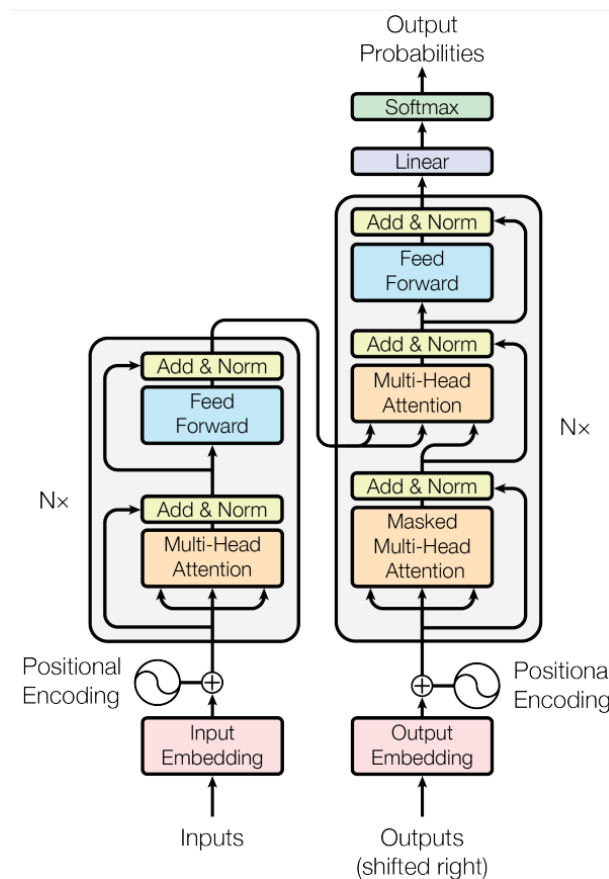


Figure 2.1: Transformer Architecture as presented in the original paper [Vas+17]. This diagram illustrates the various layers of the architecture within the Encoder and Decoder, demonstrating the flow of data through each component.

2.2.1 Encoder

For each token an embedding vector is computed, which is a very high-dimensional (currently at around 256 to 4096 dimensions) numerical vector representation of that token, that captures its semantic meaning.

A key component of the encoder is the self-attention mechanism. Self-attention enables the model to consider the entire sequence when encoding each token, allowing it to weigh the relevance of other tokens in the input sequence dynamically. For each token, the self-attention mechanism computes attention scores that determine the influence of all other tokens in the sequence. As a result, the generated embedded vector for each token not only represents the token alone but also its left and right contextual influence. The encoder consists of multiple identical layers, or encoder blocks. Each encoder block contains two main sub-layers:

- **Multi-Head Self-Attention Layer:** This sub-layer allows the model to attend to different parts of the sequence from multiple perspectives or “heads”. Each head performs self-attention independently, and their outputs are concatenated and linearly transformed to provide a richer representation.
- **Feed-Forward Layer:** After the self-attention sub-layer, each token’s representation is passed through a feed-forward neural network. This layer is a simple fully connected feed-forward network applied to each position (word) in the sequence independently and identically. It consists of two linear transformations with a non-linearity (e.g. ReLU) activation in between, allowing the model to apply non-linear transformations and further refine the encoded representation.

Both sub-layers in the encoder block are followed by residual connections and layer normalization, which help in stabilizing the training and improving convergence.

2.2.2 Decoder

The decoder has a similar structure and therefore works quite as similar as the encoder and can also be used for similar tasks. The decoder uses multiple decoder blocks, but has two additional sub-layers per block as compared to the encoder block. In the transformer’s architecture the decoder’s role is to generate the output sequence based on the encoded representation from the encoder (referred to as cross-attention). This is done auto-regressively, which means that the generated computed feature-vector, which holds information about the input sequence will be transformed by the language modelling head mapping into the next probable following word, which then will be added to the input text and is then fed back into the decoder. The most important difference to the encoder is the masked multi-head self-attention.

- **Masked Multi-Head Self-Attention Layer:** Since the decoder cannot predict future words based on information not yet generated, it only attends uni-directional

to the previously generated tokens in the output sequence. Therefore only the left context is used and the right context is masked. Masking the right context is of course only applicable for left-to-right written languages e.g. languages in european and american countries that utilize the latin script.

[Ras24; Vas+17]

2.3 BERT

BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained language representation model introduced by Devlin et al. in 2019 ([Zha+20]). Its based on the Transformer architecture from [Vas+17] but instead of using in contrast to using both, an encoder and a decoder as in the original transformer, BERT only utilizes the encoder component. Consequently, unlike other large language models (LLMs), BERT cannot predict new tokens and thus is not suitable for text generation. Instead, it achieved state-of-the-art results in tasks such as text classification, sentiment analysis, and named entity recognition. The attention scores are computed using queries, keys, and values derived from the input embeddings.

2.3.1 Embeddings

The three matrices in BERT—token embeddings, segment embeddings, and positional embeddings are generated as part of the model’s training process.

For each unique Token ID (i.e. for each of the 30,522 words and subwords in the BERT Tokenizer’s vocabulary), the BERT model contains an embedding that is trained to represent that specific token. The Embedding Layer within the model is responsible for mapping tokens to their corresponding embeddings. Before a string of text is passed to the BERT model, the BERT Tokenizer is used to convert the input from a string into a list of integer Token IDs, where each ID directly maps to a word or part of a word in the original string. In addition to the Token Embeddings described so far, BERT also relies on Position Embeddings. While Token Embeddings are used to represent each possible word or subword that can be provided to the model, Position Embeddings represent the position of each token in the input sequence. The final type of embedding used by BERT is the Token Type Embedding, also called the Segment Embedding in the original BERT paper. One of the tasks that BERT was originally trained to solve was Next Sentence Prediction. That is, given two sentences A and B, BERT was trained to determine whether B logically follows A.

BERT introduces two pre-training objectives, the masked language model objective (MLM), and the next sentence prediction objective (NSP).

- **Masked Language Modeling (MLM):** 15% of the words in a sentence are randomly masked, and the model is trained to predict these masked words based on the
-

context provided by the other words in the sentence. This enables BERT to learn bidirectional representations.

- **Next Sentence Prediction (NSP):** To understand relationships between sentences, BERT is trained on pairs of sentences. Given two sentences, the model predicts whether the second sentence is the actual next sentence in the original text or a randomly chosen one. This task helps BERT capture the coherence and context between sentences.

2.4 Fine-Tuning

After pre-training on large text corpora, BERT can be fine-tuned for specific downstream tasks using relatively small datasets. Fine-tuning involves making slight adjustments to the pre-trained model weights to better align with the target task. This process leverages robust pre-trained language representations and adapts them to meet the specific requirements of the task at hand.

One can distinguish between “Feature extraction” and “Full fine-tuning” as different types for fine-tuning. In Full fine-tuning, the entire model is trained on new task-specific data, meaning all model layers are adjusted during this process. This method is particularly advantageous when the task-specific dataset is large and significantly different from the pre-training data. Feature extraction treats the pre-trained model as a fixed feature extractor. Only the final layers of the model (or even newly added layers) are then trained on the task-specific data, while the rest of the model remains unchanged. It therefore is a more cost-effective and efficient way to fine-tune pre-trained language models.

Fine-tuning closely resembles regular training but, as mentioned, may only affect certain weights, depending on the chosen method. During fine-tuning, the loss is calculated based on the difference between the model’s predictions and the true labels using backpropagation. Gradients of the loss are then computed and used to update the model’s weights through gradient descent, iteratively refining the model to minimize the loss and enhance its performance. [Tur]

2.5 Quantization

Quantization is a technique designed to reduce the computational and memory demands of running inference using low-precision data types, such as 8-bit integers (int8), instead of the standard 32-bit floating-point (float32). This approach decreases memory storage requirements, theoretically lowers energy consumption, and accelerates operations like matrix multiplication through integer arithmetic. Moreover, it enables models to operate on embedded devices that may only support integer data types.

2.5.1 BERTScore

BERTScore is an evaluation metric that utilizes the BERT model to compare texts more semantically than traditional metrics like BLEU. It leverages the contextualized embeddings provided by a pre-trained BERT model to assess the similarity between candidate and reference texts.

The process begins by inputting both candidate and reference texts into the BERT-style model, which generates contextualized embeddings for each token in both texts. For each token, the similarity between its embedding and every token embedding in the comparison text is calculated using cosine similarity

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n \mathbf{A}_i \mathbf{B}_i}{\sqrt{\sum_{i=1}^n \mathbf{A}_i} \cdot \sqrt{\sum_{i=1}^n \mathbf{B}_i}} \quad (2.1)$$

This results in a similarity matrix where each entry represents the cosine similarity between the embeddings of a pair of tokens (one from the candidate sentence and one from the reference sentence).

The metric is computed symmetrically as follows:

For each token embedding in the candidate sentence, find the maximum similarity score with any token embedding in the reference sentence, and average these scores across all tokens in the candidate sentence to obtain precision.

Similarly, for each token embedding in the reference sentence, find the maximum similarity score with any token embedding in the candidate sentence, and average these scores across all tokens in the reference sentence to obtain recall.

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

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

Finally, the F_1 -score (an F -measure) is computed as the harmonic mean of precision and recall, providing a balanced measure that considers both the model's ability to capture relevant information and its accuracy in predicting new text equally. This works particularly well in BERTScore because the individual tokens are represented as contextualized embeddings, which means that when comparing two token embeddings and receiving similar results does not only generally mean that the tokens are quite similar semantically but also in regards to their positional context of the entire sentence.

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

[Zha+20]

2.6 BLEU-Score

BLEU-Score is a different metric I use in my thesis for comparing texts. BLEU does not evaluate and compare the semantics of the reference and candidate text but instead compares the similarity of vocabulary between them.

Let $\{y^1, y^2, \dots, y^N\}$ be the words of the reference text and $\{\hat{y}^1, \hat{y}^2, \dots, \hat{y}^N\}$

The first step is to create n-grams $G_n(y)$ for both texts. An n-gram is just a set of consecutive words of length n in a text.

$$G_n(y) = \{y_1, y_2, \dots, y_k\}$$

Next, we define the function $C(s, y)$ that counts the appearances of s as a substring in y . Now we can count n-grams of the candidate that appear in the reference text. We can compute the clipped precision by taking the minimum of the appearances of the n-gram in y and \hat{y} and then dividing by the amount of all occurrences of n-grams in \hat{y} . Therefore candidates that have the same n-gram repeating over and over again don't get a higher precision score if the same n-gram does not appear in the reference text the same amount.

$$p_n(\hat{y}, y) = \frac{\sum_{s \in G_n(\hat{y})} \min(C(s, \hat{y}), C(s, y))}{\sum_{s \in G_n(\hat{y})} C(s, \hat{y})}$$

Right now short candidate texts are more likely to get a good score although the reference text is much longer. Therefore we add a brevity penalty in order to give higher scores to texts that are closer or even longer to the reference texts' real size.

$$\text{BP}(c, r) = \begin{cases} 1, & \text{if } c > r \\ e^{(1-r/c)}, & \text{if } c \leq r \end{cases}$$

Finally, for BLEU-Score we combine the brevity penalty with the clipped precision of n-grams. We additionally add a distribution vector to weigh each p_n by w_n in order to have the opportunity to give n-grams with different n also a different impact on the overall result. Although in the end most BLEU-Scores just use a uniform distribution with $N = 4$ so that w_n always stays $\frac{1}{4}$

$$\text{BLEU} = \text{BP}(c, r) \cdot \exp \left(\sum_{n=1}^N w_n \cdot \ln(p_n) \right)$$

[Dos21; suv24]

2.7 RAG

Retrieval-augmented generation (RAG), is a technique used to improve the quality of LLM-generated responses by grounding the model on external sources, by passing the sources along with the prompt as input. LLMs are inconsistent in terms of producing same quality responses for each and every topic, since their knowledge is based on a finite amount of information, that is not equally distributed for every potential topic. But Retrieval-augmented generation doesn't only reduce the need for internal sources (continuous training, lowering computational and financial costs) but also ensures that the model has access to the most current, reliable facts. It therefore reduces the chances of generating false information and ensures that the generated content is relevant. In this thesis, I am utilizing RAG to hopefully improve (key features) from the characters described in the literature to achieve better characterizations with grounded models that utilize this external information.[Ama24]

3 Related Work

Now first of all there already has been a decent amount of approaches for automatic text summarization, which is kind of similar to creating character descriptions, although simple summaries don't include additional information for example deductions made from the behavior of that character. One of the oldest and most cited papers from 2002 belongs to "Automatic Text Summarization Using a Machine Learning Approach" from [NFK02]. It describes a summarization procedure with different compression rates based on Naive Bayes and C4.5 decision tree evaluated with precision and recall of just single word occurrences like nouns and names. Since then a lot of different techniques for summarization and text-merging and also open available datasets have been presented over the years. I will now present some of the most relevant and guided works to my thesis.

3.1 Coreference in Long Documents using Hierarchical Entity Merging

One of the most recent papers [THB24] aims to improve coreference resolution in long documents by introducing a recursive hierarchical entity merging approach.

Their method involves splitting long documents into ($n \in 2, 4, 8$) smaller segments, applying a coreference resolution model (wl-coref model(Dobrovolskii, 2021)) to each segment, yielding clusters for each entity, and then merging the entities identified in these segments using a hierarchical process. The merging process considers the embeddings of the entities to decide whether two entities from different splits refer to the same entity. This is done iteratively across all splits until a final, consolidated entity list is produced for the entire document. Since this approach is not generally dependent on the coreference model's input size, this merging approach allows coreference resolution in texts of arbitrary length.

The approach was tested on English datasets like OntoNotes and LitBank, as well as German datasets like TüBa-DZ and DROC, along with two specific German texts. The results inter alia achieved SOTA results for long document processing as the German novel "Effi Briest" attained a CoNLL-F1 score of 54.17 and more impressively a drastic LEA score of improvement from 15.92 to 39.12 while maintaining run-time efficiency of running under 2 min on a single A6000 GPU.

3.2 Evaluating Character Understanding of Large Language Models via Character Profiling from Fictional Works

The next paper was released this year during the completion of my thesis. The title is "Evaluating Character Understanding of Large Language Models via Character Profiling from Fictional Works" and it introduces the CROSS dataset, which is a dataset, that consists of expert-generated character profiles organized into four key dimensions: attributes, relationships, events, and personality.

The authors test three distinct summarization methods. Hierarchical Merging, which uses a straightforward, zero-shot prompt technique to combine summaries at lower levels into higher-level summaries. The initial level contains basic segments of the book, while subsequent levels summarize increasingly larger portions of the content. Incremental Updating is a method, that follows a similar approach but instead of creating a hierarchy, it compresses the text as the book is sequentially processed from start to finish. At any point during the summarization process, the summary only includes content up to that specific point, with the remaining content still to be summarized. The final method, summarizing in one go, utilizes LLMs with large context windows, like GPT-4-Turbo, which can process up to 128,000 tokens to generate summaries for smaller texts.

Evaluation is conducted through intrinsic assessment (Factual Consistency Examination (FCE)) and extrinsic assessment (Motivation Recognition (MR)). For the FCE Consistency Score, the authors use Llama3 to assign a score on a scale from 1 to 5, measuring the factual consistency between the reference profiles and the summaries generated by the LLMs. The Llama3 evaluation shows a high Pearson correlation with a subset of 50 randomly selected, human-evaluated samples, indicating it is a reliable evaluation metric.

Motivation Recognition (MR) is assessed by having GPT-4 generate questions for key characters. Given the character's name, the profile defined by the four dimensions, a character's decision, a question about the motivations behind the decision, and a set of potential answers, the LLMs must identify the correct answer that accurately reflects the character's motivation.

The results demonstrate that while LLMs can generate adequate profiles, challenges remain in achieving consistency and completeness. The summarizing-in-one-go method achieves the highest consistency scores across all dimensions, likely due to its ability to process the entire content of a book at once, preserving narrative coherence and minimizing information loss.[Yua+24]

3.3 Project Gutenberg

Project Gutenberg, founded in 1971 by Michael S. Hart, is one of the oldest and most extensive digital libraries, aimed at providing free access to a vast collection of over 60,000 eBooks (currently accessible at <https://www.gutenberg.org/>). Hart's initiative began

with the digitization of the United States Declaration of Independence, setting the stage for the project’s goal of democratizing access to literature and cultural works. Named after Johannes Gutenberg, the inventor of the printing press, Project Gutenberg echoes his mission of making written works widely accessible. The Project Gutenberg Literary Archive Foundation, a non-profit organization, oversees the project’s administration, legal issues, and fundraising efforts so that it can continue expanding its library and provide free access to a vast collection of eBooks without any legal repercussions.[Pro23]

3.4 LISCU

A recent paper from 2021 [Bra+21] presents a dataset called LiSCU (Literary Summaries with Character Understanding), a dataset designed to advance research in character-centric narrative understanding. The LISCU- dataset is a new dataset of literary pieces and their summaries paired with descriptions of characters that appear in them. The paper leverages this dataset for two primary tasks: character identification, where the goal is to identify a character’s name from an anonymized description, and character description generation, which involves generating a description for a given character based on a literature summary.

To manage the potential issue of summaries exceeding model input limits, the authors compare two truncation methods: length truncation, where the summary is simply cut off after a certain point, and coreference truncation, where SpanBERT is used to identify and prioritize sentences mentioning the character. For the task of character description generation, they experiment with multiple models, including GPT-2, BART (which has the double maximum input length of GPT-2), and Longformer, which supports inputs up to 16,384 tokens.

The authors evaluate the generated descriptions using BLEU, ROUGE, and BERTScore metrics. The study concludes that length truncation performs better, as key character-related information tends to appear earlier in summaries. In contrast, coreference truncation underperforms, primarily due to errors in coreference resolution impacting its effectiveness. LISCU is a dataset that is quite relevant to my thesis.

3.5 Dated Data: Tracing Knowledge Cutoffs in Large Language Models

The actual date up to which a language model retains knowledge for specific topics often differs from the reported cutoff provided by LLM creators. The authors of this paper [Che+24] identify this as potentially leading users to receiving outdated information with two main causes for these discrepancies. First, temporal biases in the CommonCrawl dataset, a widely used dataset for training LLMs, result in older data being included in newer dumps and therefore causing temporal misalignment. Second, challenges in deduplication, particularly with semantic duplicates (different versions of the same content) and

lexical near-duplicates, contribute to the inconsistencies in knowledge cutoffs.

To address these issues they propose to estimate effective cutoff dates by probing LLMs across different versions of data to analyze long-spanning datasets, such as Wikipedia and New York Times articles, in order to observe how the model’s prediction confidence (measured by perplexity) changes over time.

3.6 Methodological Approach

With this exploration in mind, I will briefly talk about the methodology focus of the upcoming experiments. Unfortunately, all the other datasets to me available at the beginning of my thesis did not include any character descriptions except for the LISCU dataset, but I was interested in the creation process on a new dataset, that can be used for similar tasks in the future. The dataset is not an extension of LISCU but a more franchise-oriented approach, that can be used to explore the model’s pretraining knowledge and to identify its behavior in different genres. The dataset can also be extended vastly since the amount of fandom articles, that are available is enormous. I will go into more detail about the creation in Chapter 4. The potential options for passage retrieval in my thesis for RAG included coreference resolution, word embeddings and model finetuning. Under the guidance of my supervisor, I prioritized creating the dataset and experimenting with different prompt wordings, followed by running baseline experiments on the new dataset. If time permits, I was to proceed with embeddings and model fine-tuning.

While the wording of LLM prompts can appreciably impact a model’s performance, I expected passage retrieval methods to enable still larger performance gains. We test two approaches for retrieving relevant passages from the original stories: (a) a method I refer to as **base line retrieval** which solely relies on string matches of character names and (b) a method which I call **passage embedding retrieval**, that instead relies on embedding similarity.

In method (a) **baseline retrieval** I filter the novels for sentences containing certain keywords. However, simply finding all sentences that mention a character’s name is insufficient for a comprehensive description. Critical information about the character may be present in sentences that do not explicitly mention their name but refer to them indirectly. Consequently, important details can be missed using this technique, and additionally, this approach might include too much unnecessary information, especially for main characters. Nevertheless, it is important to have a baseline confirmation of the functionality and improvement of the overall methods used, besides being a primitive method. From now on, I will refer to this kind of passage retrieval as **baseline retrieval**.

For method (b) **passage embedding retrieval** I split the novels into smaller chunks of roughly equal length and fed them into a model, tensor representations can be generated for each chunk. This allows the identification of sections that meet certain criteria, such as more accurately describing a particular character. I will refer to this method as **passage**

embedding retrieval.

4 Gathering of Literature

Unfortunately, there's barely any open-source collection of literature with characterizations available. Examples like "Romeo and Juliet", "Moby Dick", "Frankenstein" or "Alice's Adventures in Wonderland" are rare cases where enough fandom exists to create accessible and reviewed content. In most other instances, it seems too risky to use public-domain literature, as these collections predominantly consist of less popular books with minimal fanbase and related content. Popular literature, with its larger online presence, results in more detailed and reviewed community-generated content, such as characterizations and summaries, which are valuable as reference points for my generated characterizations, which is why I will mostly rely on non-open source literature for this thesis.

During the process of using Wikidata, a free and open knowledge database, to query characters and filter personal descriptions from books, I discovered that many of these descriptions contain references to articles from fandom.com, the world's most popular open-source wiki platform for fan-related content. Initially, I planned to query Wikidata for all characters linked to Fandom articles to gather literature with the most comprehensive fandom articles. However, I realized that not all character descriptions in Wikidata include Fandom article links. Some character descriptions are missing Fandom article URLs, making it insufficient to rely solely on Wikidata for content. Additionally, there are instances of multiple articles linked to one character. Some articles are in different languages, while others are older versions or from different universes within the same saga. In most cases, I was able to choose to use the newest, longest English version but this was not always possible. For example, when fetching Dune character fandom articles, I had to manually sort out some characters. The Dune fandom includes characters from the "Dune Encyclopedia" and "Expanded Dune", as well as from the original "Dune" by Frank Herbert. This overlap made it problematic to compare information about the same character in different contexts, especially when relevant information might not be available across all contexts.

In the end, I used multiple methods. First, I queried Wikidata to quickly obtain a large number of characters, then manually deleted duplicates and added additional characters with their corresponding fandom URLs by hand. I found that the fetched articles varied significantly in length, requiring me to cut some of them down so they were roughly the same size. I achieved this by sequentially removing paragraphs from the bottom of each original article until they reached the desired length. Truncating the end of articles yields better results than truncating the beginning, as it is unlikely that the ending contains

any essential information about the character that has not been mentioned before. Since readers of this thesis might not have access to all the non-open-source literature I used, I aimed to minimize the number of sources to make the results easier to replicate and verify. Ultimately, I was able to obtain character data for 570 characters from six franchises in total. All of the books contain text decorations and structural elements such as chapters, sections, and page numbers, which remained present after converting the PDFs and text files and loading them into memory. These elements had to be manually filtered out using regular expressions before further processing. All the results are linked in the appendix.

Book	Amount
Harry Potter	157
Dune	95
Twilight	71
The Lord of the Rings	106
The Hitchhiker's Guide to the Galaxy	82
The Hunger Games	29
Total	570

Figure 4.1: Number of Fandom Articles included in our Dataset for each Work

Franchise	Book
Harry Potter	The Sorcerers Stone The Chamber of Secrets The Prisoner of Azkaban The Goblet of Fire The Order of the Phoenix The Half-Blood Prince The Deathly Hallows
Dune	Chapterhouse_ Dune Children of Dune Dune Dune Messiah God Emperor of Dune Heretics of Dune
Twilight	Twilight New Moon Eclipse Breaking Dawn
The Lord of the Rings	The Fellowship of the Ring The Two Towers The Return of the King
The Hitchhiker's Guide to the Galaxy	The Hitchhiker's Guide to the Galaxy
The Hunger Games	The Hunger Games

Figure 4.2: List of Individual Novels per Franchise Utilized for Passage Retrieval in the Experiments

4.1 Querying Wikidata

As previously mentioned, Wikidata provides a way to directly retrieve information from its database through HTTP requests. To do so, one must formulate a SPARQL query. SPARQL has emerged as the standard semantic query language for databases that store their data in Resource Description Framework (RDF). It allows for querying required and optional patterns, conjunctions, disjunctions, supports complex queries with nested queries and then can return results in multiple formats including XML, JSON, and CSV, making it highly flexible for integration with different systems. RDF represents data as a directed graph composed of triplet statements: subject, predicate, and object. These triplets are structured by the database designer to reflect the underlying data and their relationships.

To get a more practical understanding, I will now briefly explain a query that I built to retrieve character names with their corresponding fandom URLs given a fandom page name (in this case Lord of the Rings).

```

1  SELECT
2  ?item
3  ?fandom
4  ?fandomStatement
5  ?characterName
6  WHERE {
7  ?item wdt:P31 wd:Q3658341.
8  ?item p:P6262 ?fandomStatement.
9  ?fandomStatement ps:P6262 ?fandom.
10 ?fandomStatement pq:P1810 ?characterName.
11 BIND(STRBEFORE(?fandom, ":") AS ?firstHalf).
12 FILTER (?firstHalf = "lotr").
13 SERVICE wikibase:label { bd:serviceParam wikibase:language "en". }
14 }

```

Figure 4.3: SPARQL query to retrieve selected items, fandoms, fandom statements, and character names filtered by the "lotr" fandom from wikidata.

Below the SELECT clause, we define some variables to store parts of the triples specified in the WHERE clause. This query retrieves entities that are instances (P31) of literary characters (Q3658341) and stores the statement node for the Fandom Article ID (P6262) in the `?fandomStatement` variable. From this statement node, we extract the fandom object (P6262) and store it in the `?fandom` variable. We also extract the character name (p1810) from the statement node and store it in the `?characterName` variable. Finally, we filter the results to include only those where the fandom URL starts with the desired prefix "lotr". Note that I used different qualifiers in the query, such as p, ps, or pq, to access different desired parts of the predicate, such as only the statement node, the actual value, or the qualifier value.[W3C08; Wik24]

4.1.1 Scraping Fandom Articles


Nearly every character article from fandom.com starts out with a quote that mostly belongs to that character and describes him well. Then there is a general short summary, that gives the most important aspects of that character followed by an agenda for all the specific content that follows right after. Usually, the agenda contains topics such as “Biography”, “Character” and mostly also “Gallery” and “References”. The agenda is different for every character. Fandom articles do not have to be about characters but can also be about anything else e.g. places, traditions, ethnics, artefacts or ideas in the same universe that are interesting to the writers. Any article is at least peer-reviewed twice before publishment and anyone after that can review it and ask for changes if they detect any faults. The Figure 4.4 illustrates the structure and appearance of such an article.

"He will never be rid of his need for it. He hates and loves the Ring, as he hates and loves himself. Sméagol's life is a sad story. Yes, Sméagol he was once called. Before the Ring found him. Before it drove him mad."

—Gandalf describing Gollum to Frodo

Gollum, originally named **Sméagol** (or **Trahald**), was a **Stoor**, one of the three early **Hobbit**-types in the **Third Age**. His given name of Sméagol should be pronounced as *smay-ah-gol*.

By possessing the **One Ring**, his life extended centuries beyond his nature, and he became deformed and twisted in body and mind by the corruption of the Ring. He pursued the Ring for many years after **Bilbo Baggins** found it in **Gollum's cave**, within the **Misty Mountains**, and took it away. Inadvertently, Gollum would play a vital role in the ultimate **Quest of the Ring**. The name *Gollum* was derived from the sound of his gurgling, choking cough.



"We Are Lost" by [Dean Vito](#)

Follow on IG | TikTok | Join Fan Lab | Check out Fandom Quizzes and cha

Gollum / Sméagol

Biographical information

Other names	Trahald, Slinker, Stinker, Shelob's Sneak
Titles	Ring-bearer
Birth	TA 2430

Contents [\[hide\]](#)

- Biography
 - Early life
 - Life under the Misty Mountains
 - Search for the Ring
 - The War of the Ring
 - Death
 - Legacy
- Character

The War of the Ring

Gollum met and started following the Fellowship of the Ring in Moria, and was spotted and heard by [Frodo](#) on several occasions. On [January 15, 3019](#), the Fellowship was divided when Gandalf disappeared while fighting a [Balrog](#) (though he later returned). Gollum continued trailing the remaining members. It is unknown how he crossed the broken [Bridge of Khazad-dûm](#), but he came with them to [Lórien](#) without their knowing. Gollum followed their boats down [Anduin](#) (floating on a log) to [Nen Hithoel](#) and pursued Frodo and [Sam](#) across the [Emyn Muil](#) when they struck out on their own towards Mordor. Gollum followed them, but after a confrontation (in which he bit and nearly strangled Sam for the Ring), Frodo subdued him and threatened to kill him with [Sting](#), the Elvish blade that Gollum had apparently recognized by its former owner - Bilbo. Sam tied an [elven-rope](#) around Gollum's neck for a leash, which inflicted great pain on Gollum. Taking pity, Frodo made Gollum swear to help them. Agreeing to the oath, Gollum swore by the "Precious" itself as the Ring was treacherous and would hold Gollum to his word, so Frodo released him to show them the way to Mordor. The unlikely company, guided by Gollum, made its way to the [Black Gate](#) of Mordor.



Gollum on the hunt for Bilbo, by [Eric Velthagen](#)

Figure 4.4: Sample Fandom Article for the Character "Gollum" from "The Lord of the Rings" Fandom Site (<https://lotr.fandom.com/wiki/Gollum>)

5 Experiments

All my experiments have been conducted partially on my own computer but also over SSH on a remote server from the LT group at the University of Hamburg. This was mainly due to accessing a better GPU like the NVIDIA RTX A6000 for doing more computational intensive work such as prompting large language models (LLMs) and generating embeddings.

5.1 Baseline Experiment

For my first experiment, I formulated four prompts with slightly different wordings to observe how varying prompts affect the outcomes of the LLM. For each prompt, I tested two versions: one with additional passages from the literature providing information about the character, and one without such information, requiring the model to solely rely on its training data. All the eight raw prompts 5.1 contain tags. These tags will be interpreted as follows. “[INST]” and “[\INST]” mark the start and end of each query instruction. “{character}” and “{book}” will be replaced with the real character name and book title. “{passages}” marks the spot where a collection of retrieved passages for the given character from the corresponding book will be passed into. Notice that by “real character name” I am referring to the canonical name displayed in the fandom article. Only few characters may have one or more aliases, which are mostly observed in already quite popular characters. Retrieving these aliases and adding them to the prompt seems to be too high effort-low reward and is therefore left out.

As you can see, P_2 is more specific, requesting the style of a fandom article, whereas P_3 is less precise, asking only for an overview without specifying a particular format. The last prompt P_4 is similar to P_1 but is intentionally faulty by missing characters. These different prompts are used to determine the overall effects of various prompt wordings and faulty instructions on the language model.

In the baseline experiment, I defined the baseline retrieval as follows. The additional information from the book is selected by filtering for every sentence in which the character’s name occurred at least once. Since the number of tokens might exceed the maximum input size of the LLaMA model, I removed every n -th sentence, where n is calculated in such a way that the query size fits perfectly. Additionally, to account for characters being more likely to be introduced in the first sentences they are mentioned, an additional hyperparameter $\alpha \in [0, 1] \subset \mathbb{R}$ to control the cutoff is used. This cutoff represents the percentage of

Prompt	Instruction
P_1^z	"[INST]Write a summary about the character {character} in the book {book}[/INST]"
P_1^r	"[INST]Write a summary about the character {character} in the given text passages: \n {passages}[/INST]"
P_2^z	"[INST]Write a summary in the style of a fandom article about the character {character} in the book {book}[/INST]"
P_2^r	"[INST]Write a summary in the style of a fandom article about the character {character} in the given text passages: \n {passages}[/INST]"
P_3^z	"[INST]Provide a concise overview of the character {character} from the book {book}[/INST]"
P_3^r	"[INST]Provide a concise overview of the character {character} based on the following excerpts: \n {passages}[/INST]"
P_4^z	"[INST]rite sumary bout thee cara cter {character} of th book {book}[/INST]"
P_4^r	"[INST]rite sumary bout thee cara cter {character} bsd th flowing excerpts: \n {passages}[/INST]"

Figure 5.1: Structures of All Prompts Used for Zero-Shotting and Passage Retrieval in the Baseline Experiment

relevant sentences (with character name occurrences) to which every sentence with name occurrence will be taken, so the rule of taking every n -th sentence only affects sentences after the cutoff. Overall the passage retrieval for this experiment R_{base} works as follows. Let $S = \{s_i \mid 1 \leq i \leq k\}$ be the set of size k which contains all relevant sentences containing the character and l be the maximum input size of the Llama query. We first define a function $S_t(a, b) = \{s_{ti} \mid a \cdot k \leq ti \leq b \cdot k\}$, that enables a range selection of sentences with a lower and upper limit and a parameter t for the stepsize. If we now choose our n the right way

$$n = \begin{cases} \lfloor \frac{k-\alpha k}{l} \rfloor & \text{if } k - \alpha k > l \\ 1 & \text{otherwise} \end{cases}$$

we can write R_{base} as

$$R_{base} = S_1(0, \alpha) \cup S_n(\alpha, 1)$$

. I utilized the Mixtral7B model [Hug] with the smallest quantization level ($Q2_K$ weights) to process the prompts to achieve the quick responses. For evaluation, I used BLEUScore and BERTScore to compare the generated results against manually written articles from fandom.com.

To analyze the results, I decided to use boxplots, T-tests, and Spearman's correlation. To quickly summarize, a paired T-test compares the means of two related groups to de-

termine if there is a statistically significant difference between those means. I used it to have quantitative proof that the results improved after the passage retrieval. A boxplot divides the data into four parts by determining the three quartiles. Additionally, the added histograms on the sides of the scatterplots allow for a better overview of the distribution of the data. The Spearman's correlation evaluates monotonic relationships between two ranked variables, which helps to identify the general tendency of the passage retrieval.

5.1.1 Results

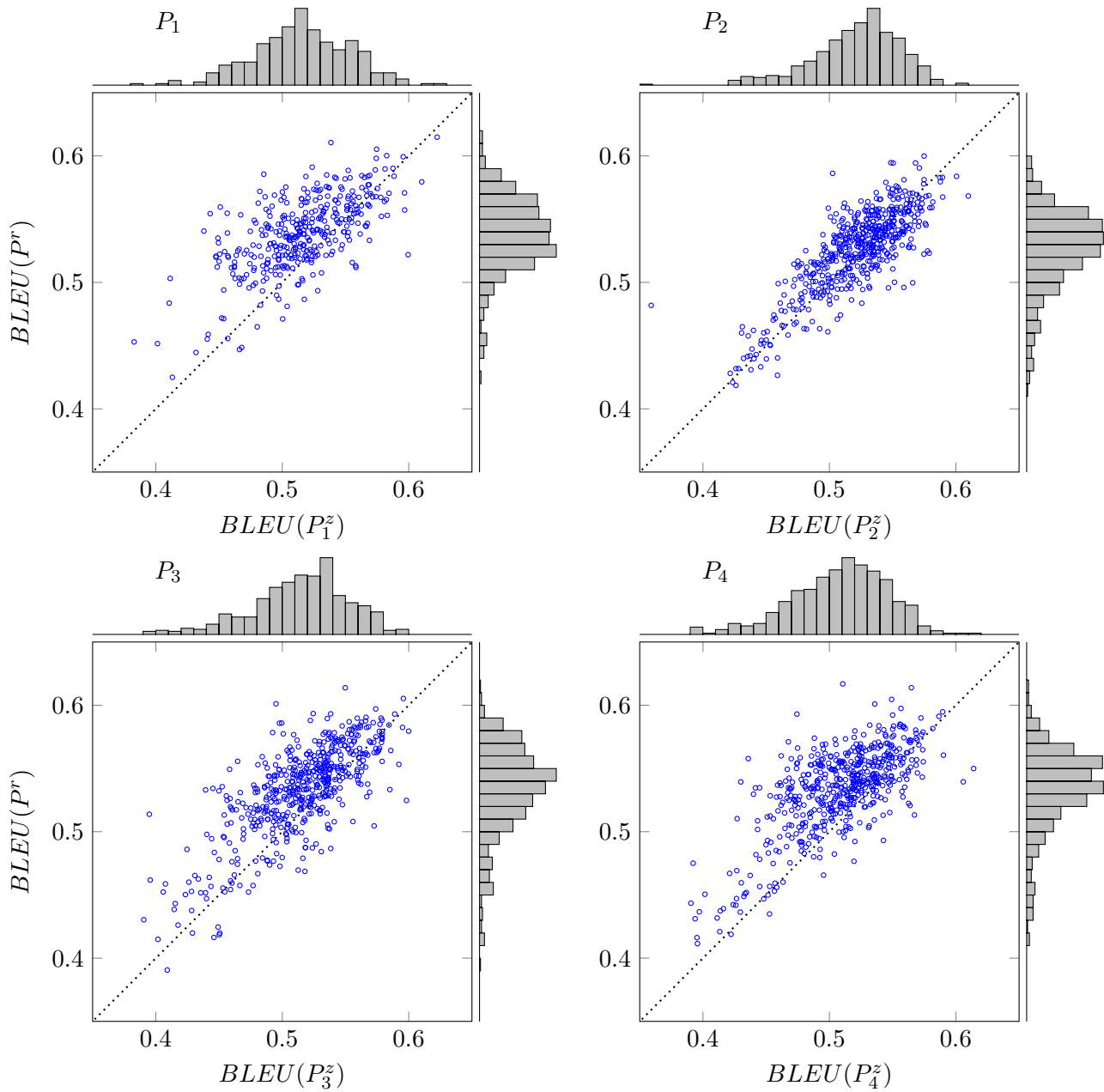


Figure 5.2: Scatterplot of BLEUScores of character descriptions generated with Mixtral7b without (P^z) and with baseline passage retrieval (P^r) plotted against each other

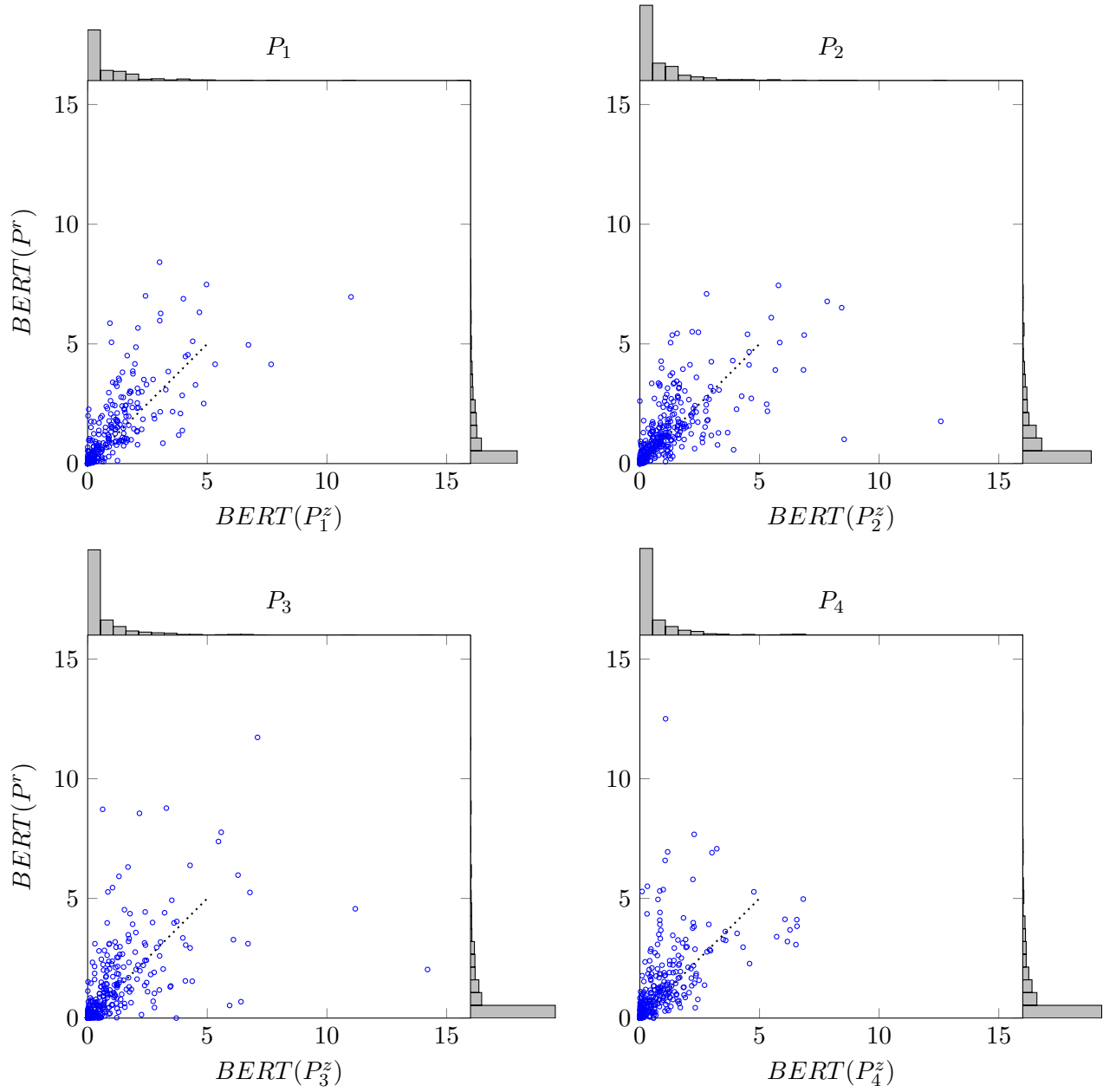


Figure 5.3: Scatterplot of BERTScores of character descriptions generated with Mixtral7b without (P^z) and with baseline passage retrieval (P^r) plotted against each other

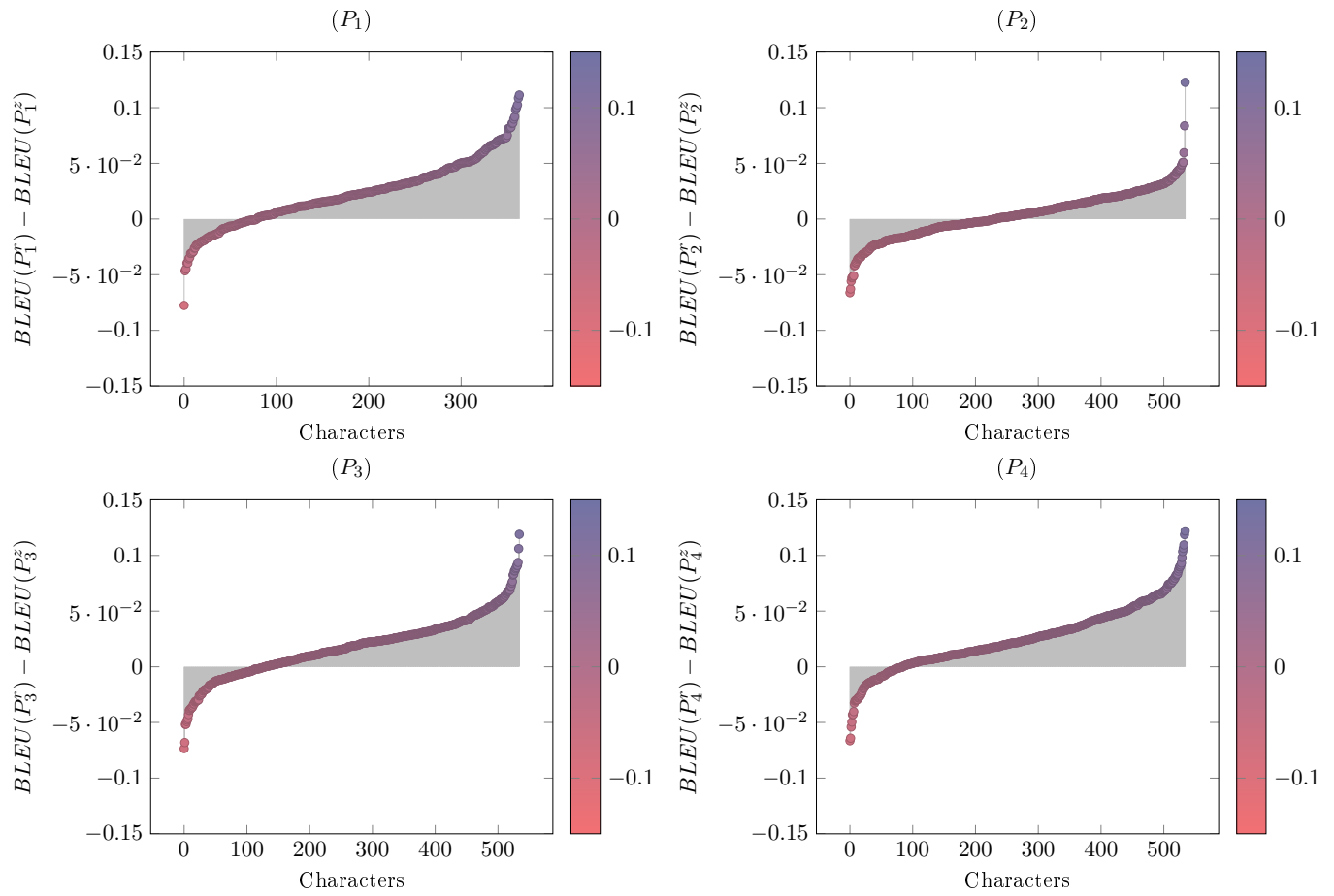


Figure 5.4: Sorted differences between BLEUScores of character descriptions generated with Mixtral7b without (P^z) and with baseline passage retrieval (P^r) for every prompt (P_1, P_2, P_3, P_4).

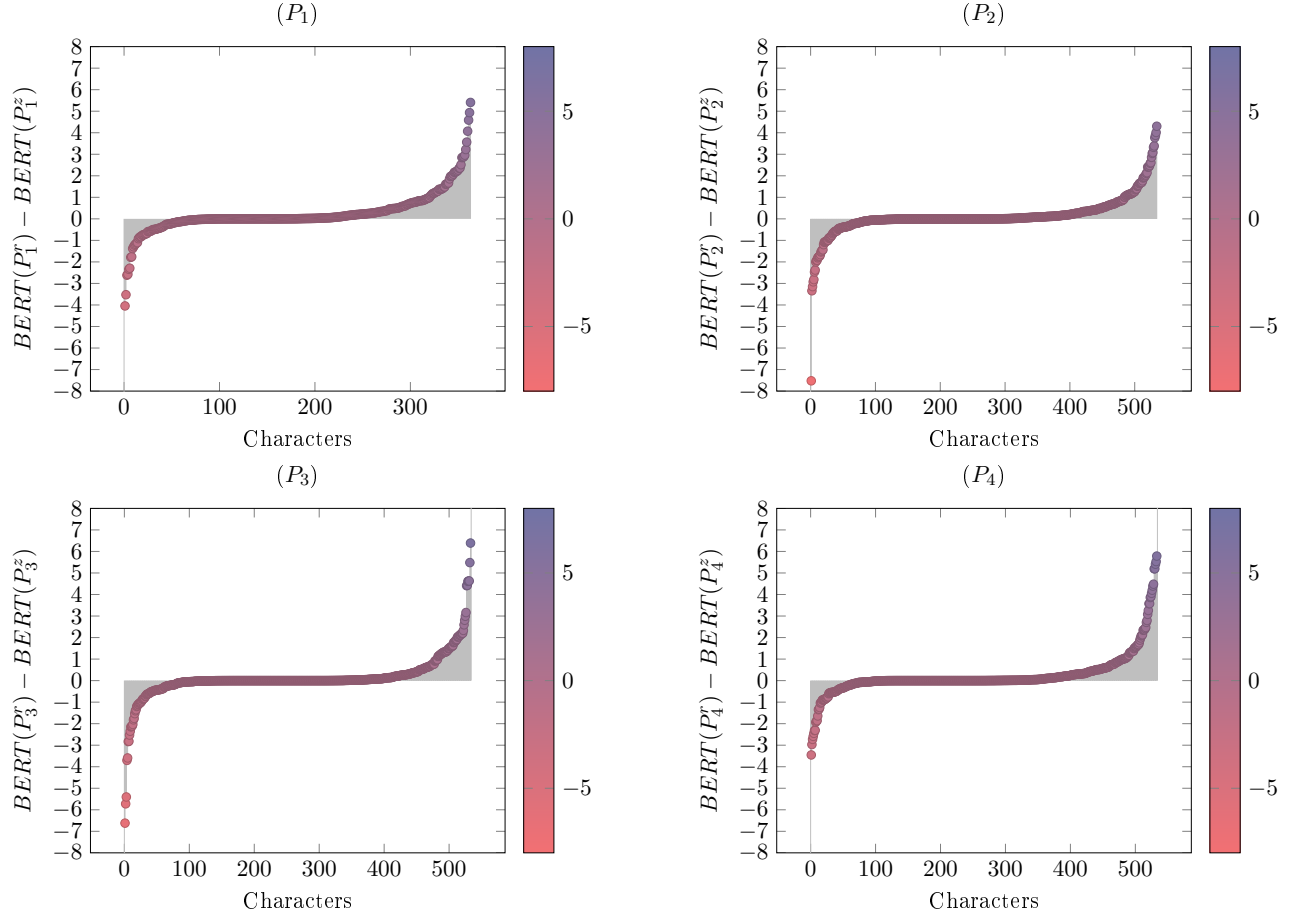


Figure 5.5: Sorted differences between BERTScores of character descriptions generated with Mixtral7b without (P^z) and with baseline passage retrieval (P^r) for every prompt (P_1, P_2, P_3, P_4).

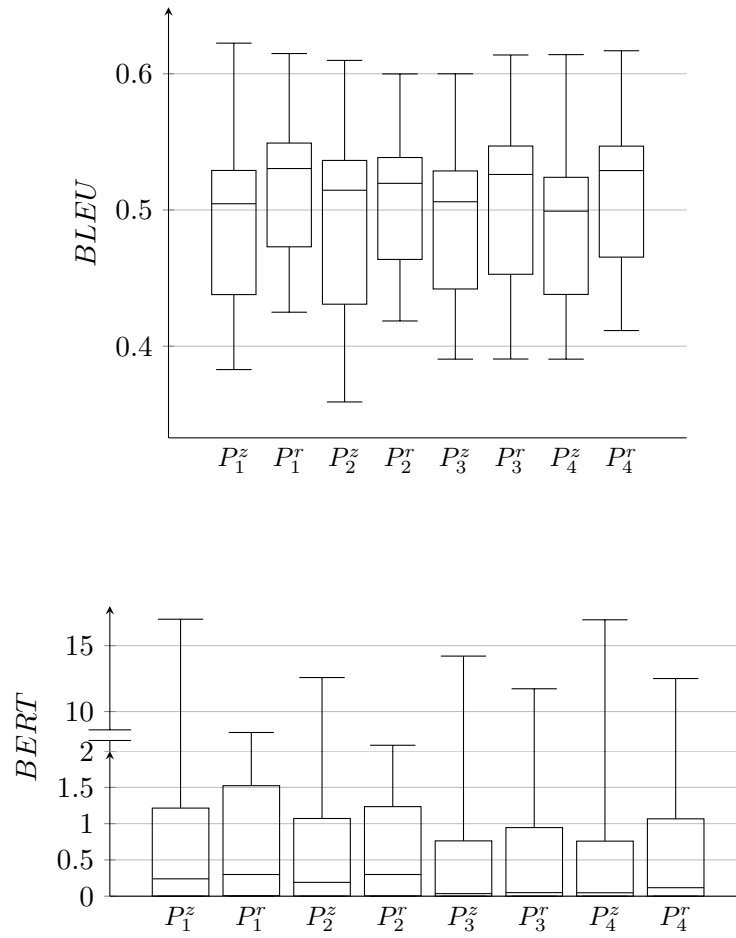


Figure 5.6: Boxplots of BLEU- and BERTScores of character descriptions generated with Mixtral7b for every prompt (P_1, P_2, P_3, P_4) with and without baseline passage retrieval.

Prompt	Heuristic	T-Test	Spearman's Correlation
P_1	BLEU	-15.0 (6.65e-40)	0.62 (3.46e-40)
P_1	BERT	-3.04 (2.56e-03)	0.91 (4.34e-142)
P_2	BLEU	-4.2 (3.08e-05)	0.74 (1.23e-95)
P_2	BERT	-2.58 (1.00e-02)	0.92 (5.12e-224)
P_3	BLEU	-15.52 (4.30e-45)	0.7 (1.76e-79)
P_3	BERT	-2.06 (3.94e-02)	0.92 (2.51e-218)
P_4	BLEU	-20.65 (4.30e-70)	0.61 (6.66e-57)
P_4	BERT	-4.0 (7.13e-05)	0.92 (2.14e-220)

Figure 5.7: T-Test and Spearman's correlation (with Corresponding p-values) of BLEU- and BERTScores of character descriptions generated with Mixtral7b for every prompt (P_1, P_2, P_3, P_4) with and without baseline passage retrieval.

5.1.2 Analysis

Having a more detailed look at the results, we can see that the T-Test scores for BLEU- and BERTScores of each prompt have all improved after passage retrieval.

The BLEUScore improvements are more drastic. P_1 , P_4 and P_3 have T-Test (5.7) values (negative values indicate an improvement in the T-Test) smaller than -14.0 with p-values in the order of magnitude of around -40. P_2 however has only slightly improved with a score of -4.2 (p-value=3.08e-05). The boxplots reveal, that the maximum values of P_1^z and P_2^z have decreased slightly in P_1^r and P_2^r , the minimum values, Q1, and Q3 have significantly increased, as observed in the boxplots. For both P_3^z and P_4^z , every boxplot quartile has improved.

For BERTScore, the improvement is not quite as obvious. In fact, the upper quartiles have a lower maximum after passage retrieval, but Q1-Q3 have improved slightly across all prompts. In other words, the distribution of BERTScore shows less variance as we introduce additional information using RAG.

The exceptional BERTScore outliers in the setup without RAG might be attributed to characters for which a significant number of training samples, closely resembling the target summary, are present in Mixtral's training data. Especially when generating summaries for main characters, Mixtral might already have a great knowledge base for that character, and relying solely on the additional passed sentences might therefore be hindering in generating a good character descriptions. Obviously the method of passage retrieval used for this experiment is not ideal, as regularly eliminating sentences could omit important context. Please note that this experiment was performed on a preliminary dataset that included some duplicates and missing character profiles. Despite these limitations, the data is still sufficient enough to show two important aspects. First, the results with passage retrieval are at least as good as, or already slightly better than, those without. The similarity of the

vocabulary has increased quite significantly whereas semantics seem only to have improved slightly, just barely scratching the statistical significance level of $p < 0.05$.

Second, the results and the different wordings in the prompts definitely have an influence on the results average and variance 5.2. Nevertheless choosing the right prompt for this task is not as simple as choosing the results with the highest score average, a low variance is even more crucial since it testifies to a higher precision and is therefore a more accurate prompt for achieving the desired output. Based on this deduction and the observations of BERT- and BLEUScore I will continue the next experiment with prompt P_1 . P_1 did not achieve the highest BLEUScore values but it for sure has the highest average while maintaining a low variance after passage retrieval. The same can be said about BERTScore. The only prompt that comes close to P_1 is P_2 . P_2 has the highest Pearson Correlation for BLEUScore that can also be easily obducted by just looking at the Scatterplots (5.2). P_2 delivers more consistent results but still does not deliver a higher BLEUScore average and higher BERTScores spread into the upper quartile than P_1 .

5.2 Passage Embedding Experiment

We will now continue with P_1 from the base experiment and apply passage embedding retrieval. Rather than selecting n sentences containing the target name, we first divide each book into chunks of approximately 1,000 character symbols. For reasons of simplicity, we employ the chunking implementation `SpacyTextSplitter` from the `langchain` library. Alternatives for normal continuous text would include the `NLTKTextSplitter` which is a slightly more resources performant text splitter but not taking any context preservation into account. Nevertheless due to the sliding window approach used by both of them and just a chunk size of 1000, the difference, if any is completely negligible. Next, we use the `e-5-mistral` model [Int] to generate embedding tensors for each chunk. We then compare the similarity scores between these tensors and an embedding tensor to a query specifically designed to meet our requirements. The chunks with the highest matching scores are retrieved and subsequently reordered to reflect their original sequence in the book. The query employed for this task is as follows: “Given passages of a book, retrieve the most relevant passages that best describe the following character {character}.”. For the text generation, I used the two current models `Llama3` (i.e. `llama3:8b-instruct-fp16`) and `Gemma2` (i.e. `gemma2:9b-instruct-fp16`). Both have a context length of 8192 tokens leaving room for about 4096 tokens for the input query and passage retrieval. I will also zero-shot the prompts without passage retrieval in order to see how the scores improved from the baseline retrieval and from `Mixtral7b`. In order to estimate the number m of chunks that can be used for passage retrieval, I conducted a small experiment on the fetched fandom articles to average out the number of words per token which resulted in a mean and median of around 0.28. Since every chunk has a length of roughly 1000 characters the possible amounts of chunks can be calculated easily as $m = \lfloor \frac{4096}{1000 \cdot 0.28} \rfloor = 14$. I decided to reduce m even further to 13 just to leave enough room.

5.2.1 Results

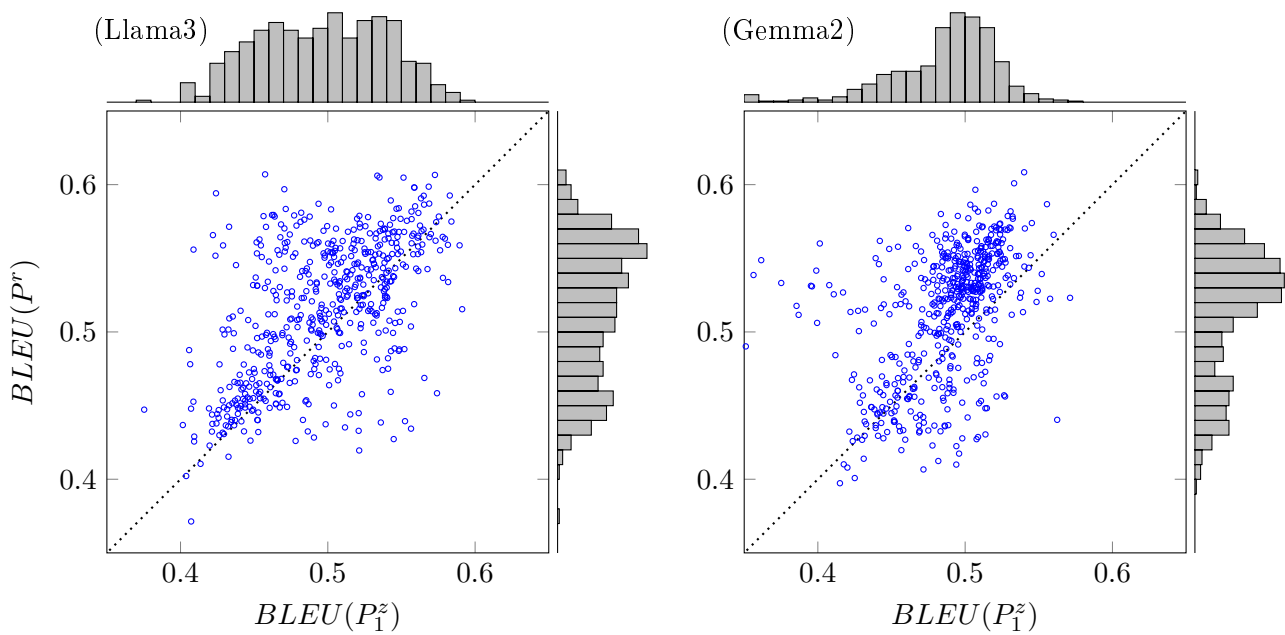


Figure 5.8: Scatterplot of BLEUScores of character descriptions generated with Llama3 and Gemma2 on P_1 without (P^z) and with passage embedding retrieval (P^r) plotted against each other.

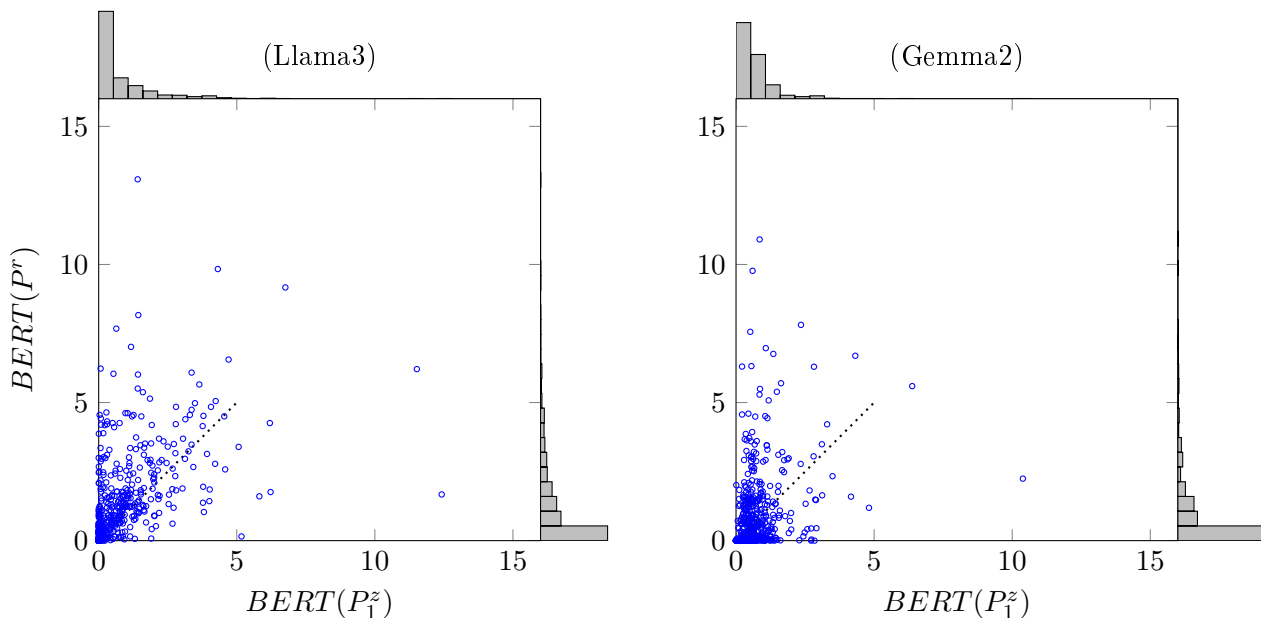


Figure 5.9: Scatterplot of BERTScore of character descriptions generated with Llama3 and Gemma2 on P_1 without (P^z) and with passage embedding retrieval (P^r) plotted against each other.

Model	Heuristic	T-Test	Spearman's Correlation
llama3	BLEU	-10.31 (6.99e-23)	0.55 (2.40e-43)
llama3	BERT	-7.42 (4.52e-13)	0.77 (1.03e-106)
Gemma2	BLEU	-14.29 (1.64e-39)	0.53 (7.09e-40)
Gemma2	BERT	-3.69 (2.43e-4)	0.56 (6.30e-46)

Figure 5.10: T-Test and Spearman's Correlation (with Corresponding p-values) after prompting Llama3 and Gemma2 with P_1 with and without passage embedding retrieval

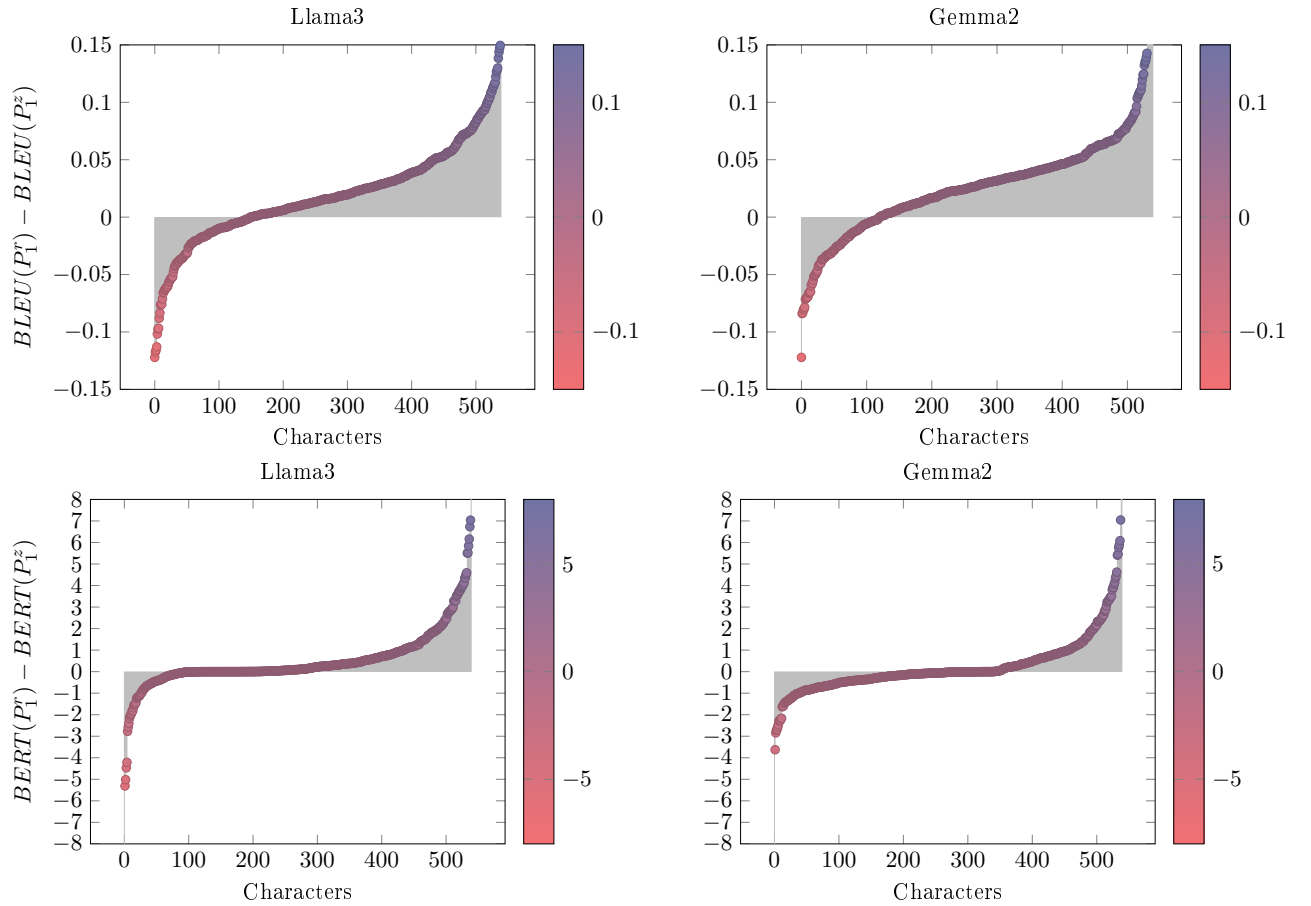


Figure 5.11: Sorted differences between BLEU- and BERTScores of character descriptions generated with Llama3 and Gemma2 of P_1 without (P_1^z) and with passage embedding retrieval (P_1^r).

5.2.2 Analysis

Looking at the results we can see that first of all there is a clear improvement for both BLEU- and BERTScore after passage embedding retrieval for both models. Although BERTScore improved too, the changes seem to be more drastic for BLEUScore. For

BLEUScore, the results generated by Gemma2 seem to have a concentrated cluster at around (0.5|0.55), whereas the results from Llama3 look more equally distributed. Unfortunately for BERTScore most of the results have not changed too much. The results from Gemma2 already have a slightly better score but the results have not improved as much as the results from Llama3. So it seems that Gemma2 already is very consistently using similar words as the fandom articles, even without passage retrieval, and has even improved stronger in BLEUScore than Llama3. At this point, I have to mention, that the version of Gemma2 I chose was not quite easy to work with. I think it is best to just show an example from the results:

```
"This is a great excerpt from Harry Potter! It seems like you've
pulled together several scenes that focus on Tonks and her
relationship with Remus Lupin. \n\nHere are some key takeaways
and observations:\n\n* **Tonks's Transformation:** We see a
stark contrast between Tonks's personality before and after the
events at the Ministry. She's lost her lightheartedness and
seems burdened by a heavy sadness. This suggests the trauma she
experienced has deeply affected her.\n* **Her Love for Lupin:**
The scene where Tonks confronts Lupin about his werewolf status
reveals the depth of her feelings for him. Despite the danger
and societal stigma, she is determined to be with him. \n* **
Lupin's Struggle:** Lupin appears conflicted and hesitant about
accepting Tonks's love. He worries about being a burden and
believes he is too old and "dangerous" for her. This internal
conflict adds another layer of complexity to their relationship
.\n* **Harry's Guilt:** Harry feels responsible for the
potential danger faced by Tonks and the other Order members. His
decision to provide his hair for the plan weighs heavily on him
, highlighting the consequences of his actions.\n\n**Overall,
this excerpt effectively portrays the emotional turmoil and
complexities surrounding Tonks and Lupin's relationship in the
wake of tragedy.** It also emphasizes Harry's growing awareness
of the sacrifices made by those fighting against Voldemort.\n\n
Let me know if you have any other questions or want to discuss
specific aspects of the excerpt further!"
```

Figure 5.12: Prompting result for the character “Nymphadora Tonks” from the Harry Potter franchise generated with P_1^r (passage embedding retrieval) in Gemma2.

Gemma2 uses a lot of markdown. The results are heavily structured with bullet points or headlines. Gemma2 also praises me at the beginning for the provided passages and asks me at the end whether I need further assistance. This character is not an exception, but just an arbitrary choice. I observed that almost every character demonstrates similar patterns. Attempts to suppress this behavior through system prompting were only partially

successful, and the quality of the descriptions noticeably deteriorated when trying to adjust it directly.

In contrast, I did not encounter similar issues with Llama3. As noted in Meta’s recent paper, "The Llama 3 Herd of Models" ([Dub+24]): “We find markdown is harmful to the performance of a model that is primarily trained on web data compared to plain text, so we remove all markdown markers.”

5.3 Method Comparison

We now have clear evidence that passage retrieval, both in the Passage Embedding Experiment and the Base Experiment, positively impacts the quality of generated character descriptions—at least within the scope of the experiment. However, it remains unclear which passage retrieval method performs better. Is it worth the computational effort to create embeddings for generating character descriptions? If so, is there a specific method or model that outperforms the others? To address these questions, we will compare the results of the two experiments conducted with prompt P_1 across different models and evaluate the differences in their outcomes.

5.3.1 Results

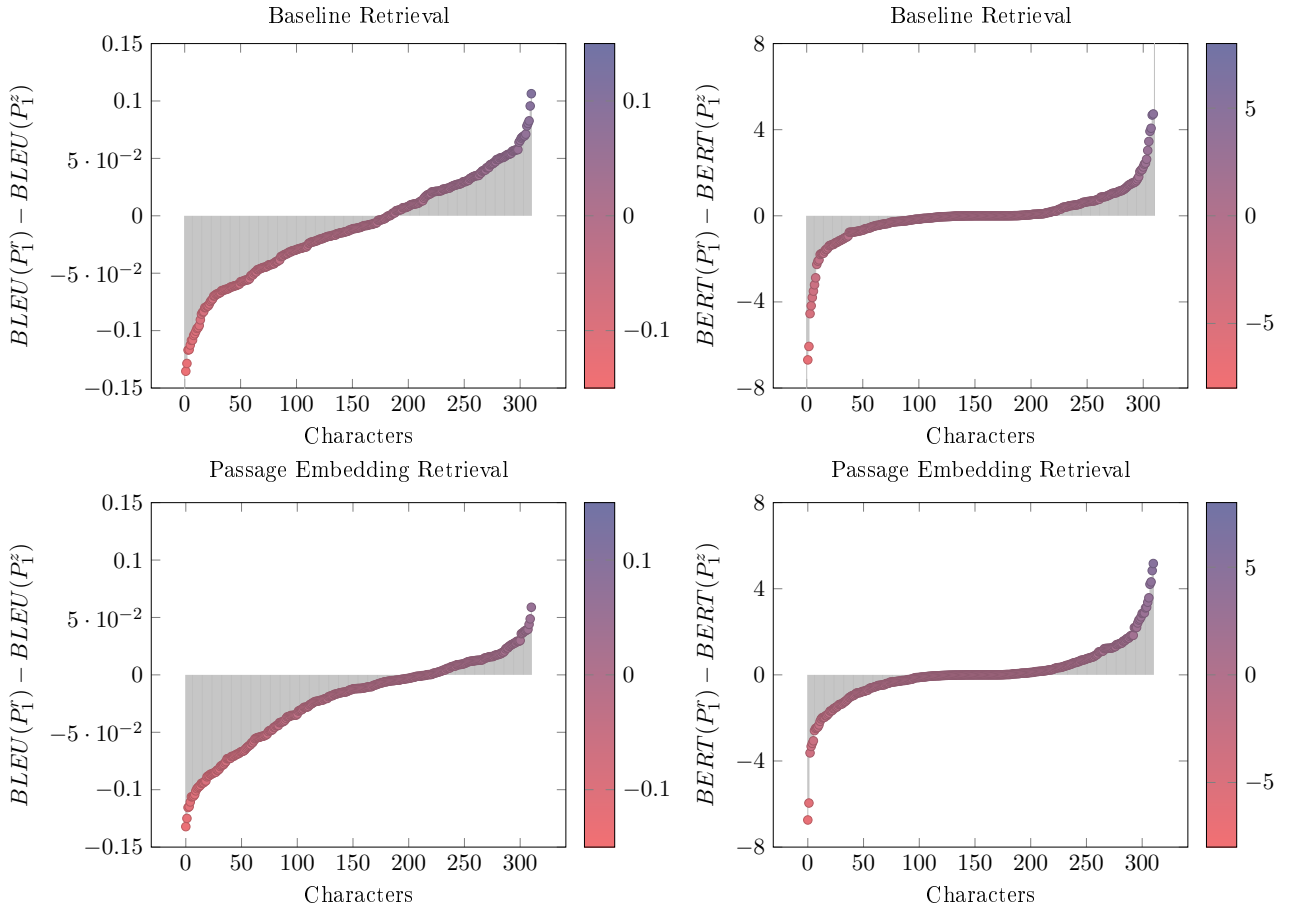


Figure 5.13: Sorted Differences of BLEU- and BERTScores for P_1 Between Baseline Retrieval on Mixtral7b and Passage Embedding Retrieval on Llama3 Before and After Retrieval

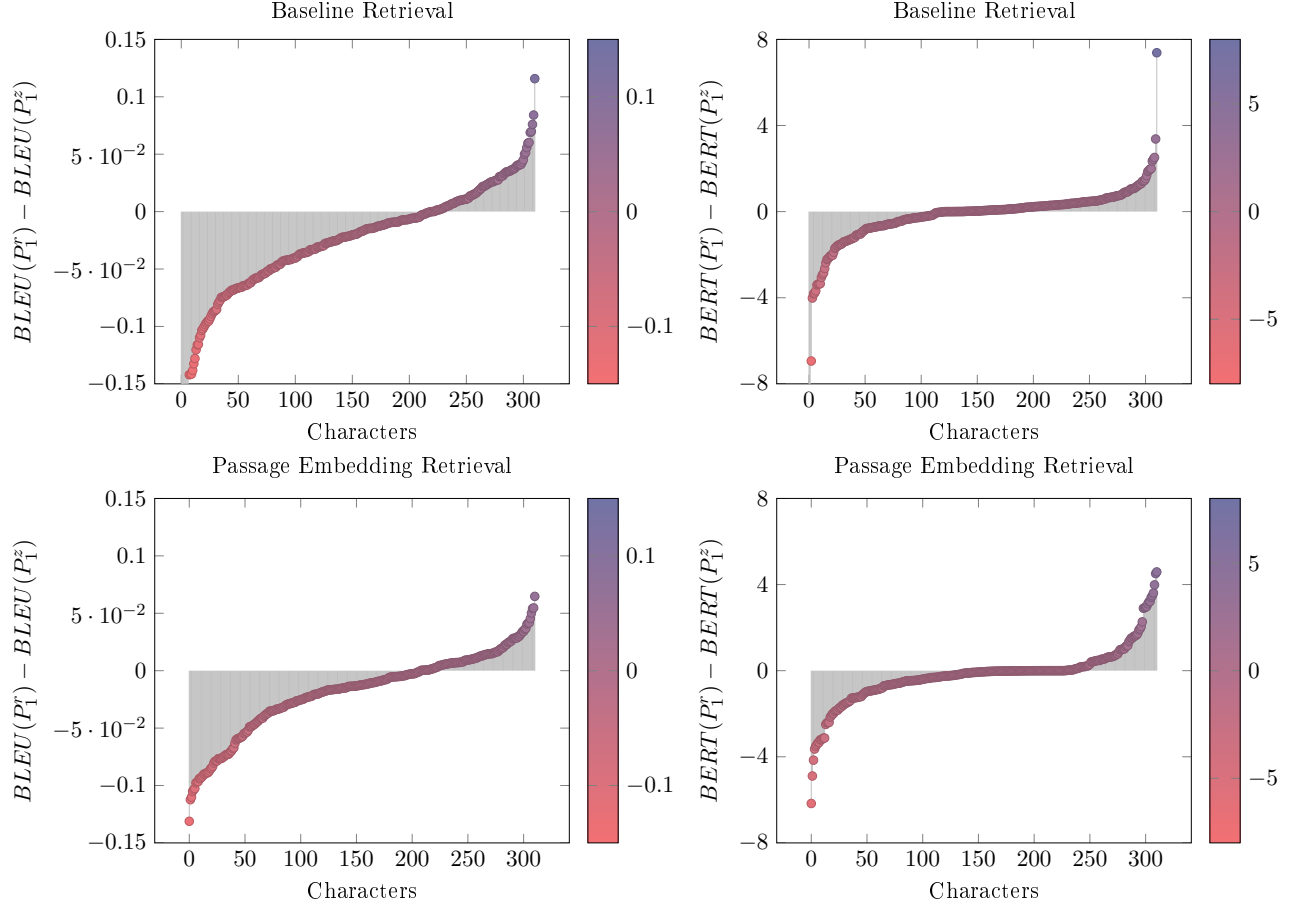


Figure 5.14: Sorted Differences of BLEU- and BERTScores for P_1 Between Baseline Retrieval on Mixtral7b and Passage Embedding Retrieval on Gemma2 Before and After Retrieval

Model	Heuristic	Retrieval	T-Test	Spearman's Correlation
Llama3	BLEU	Before	4.2 (3.48e-05)	0.33 (2.83e-09)
Llama3	BLEU	After	10.32 (1.18e-21)	0.5 (1.11e-20)
Llama3	BERT	Before	0.13 (8.93e-01)	0.52 (7.54e-23)
Llama3	BERT	After	-0.77 (4.41e-01)	0.56 (1.55e-26)
Gemma2	BLEU	Before	9.3 (2.61e-18)	0.3 (4.19e-08)
Gemma2	BLEU	After	9.11 (1.02e-17)	0.49 (2.29e-20)
Gemma2	BERT	Before	2.24 (2.57e-02)	0.48 (1.60e-19)
Gemma2	BERT	After	2.59 (1.01e-02)	0.55 (1.57e-25)

Figure 5.15: T-Test and Spearman's Correlation (with Corresponding p-values) of BLEU- and BERTScore for P_1 Using Base Retrieval and Selected Embedded Chunk Retrieval on Llama3 and Gemma2

5.3.2 Analysis

Looking at the T-Test it seems that the BERTScore stays unchanged and the BLEUScore seem to have drastically worsen. A further investigation of the results, conducted on a case-by-case basis, discovered a potential cause of the underperformance of the embedding RAG approach. In the process of retrieving fandom articles from [fandom.com](https://lotr.fandom.com/), there are instances where it is unclear which books are associated with the specific article in question. For example, the Lord of the Rings fandom (<https://lotr.fandom.com/>) includes not only the three Lord of the Rings books but also characters from "The Hobbit" and "The Silmarillion." As a result, when embedding retrieval is applied to a character name that does not appear in the specific book being referenced, the retrieval process may return passages describing a different or similar character. This poses a significant challenge for the language models used for character generation. When the retrieved passages do not correctly correspond to the intended character, the models may either fail to identify the character or suggest alternative narratives for unrelated characters.

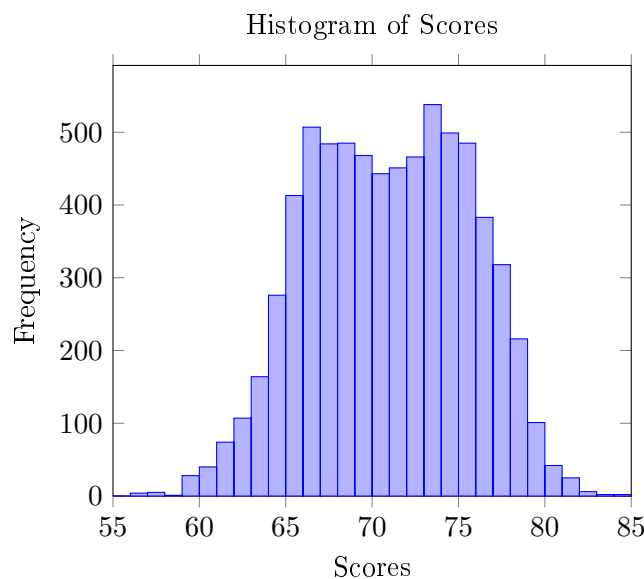


Figure 5.16: Distribution of the embedding scores of all passages used for passage retrieval computed with e5-mistral.

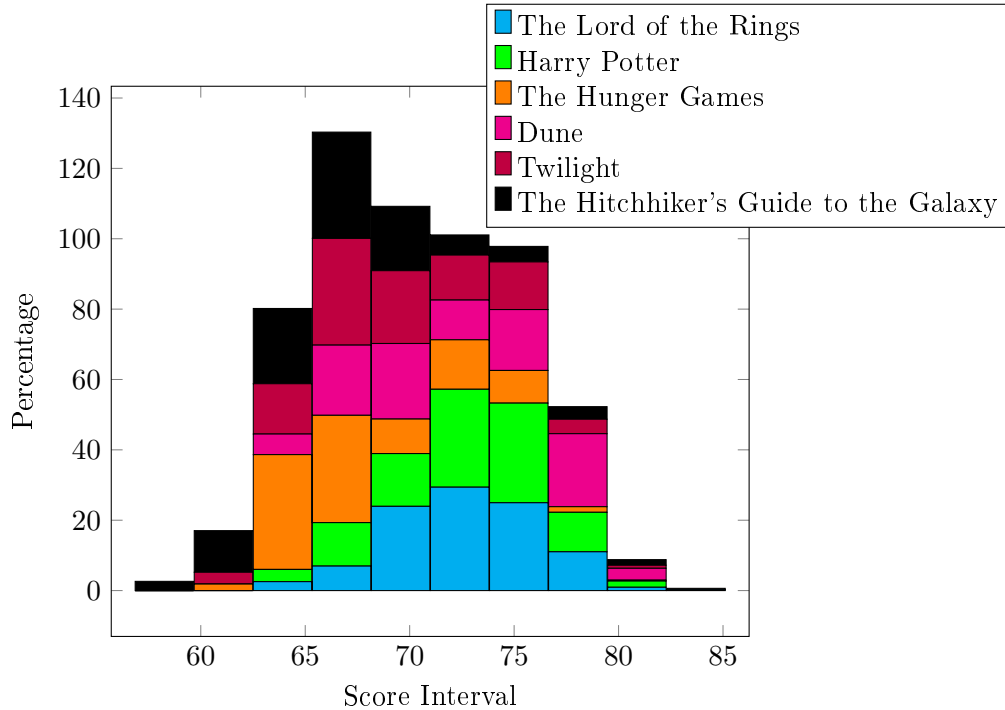


Figure 5.17: Distribution of embedding similarity scores (in percentages) for all passages used for retrieval, categorized by fandom and score interval, calculated using e5-mistral.

As illustrated in the histogram 5.16, a distinct double hump is evident, indicative of a discernible segregation in scores between passages that effectively portray a particular character and those that do not. It is clear that there is a distinct separation between the characters that are present in the book and those that are not. The number of scores in the intermediate range is relatively limited, as the likelihood of passages in a lengthy text that provide only partial information about a character and still score into the top 13 passages describing that character is relatively small.

Additionally, we now have proof that the poor passage retrieval is caused by missing novels from the entire franchise. In figure 5.17, one can see that the highest embedding scores are yielded for Dune, Harry Potter and The Lord of the Rings, which are the most complete and largest parts of my dataset. For both "The Hitchhiker's Guide to the Galaxy" and "The Hunger Games," I only included one novel each, so the datasets are only complete to around 17% and 25%, respectively. Twilight falls between them, with a dataset completeness of 67%. This correlates very well with the score observations in figure 5.17.

Given that our data is not as clean as we would have preferred, I will conduct a further investigation with a new T-test, including only characters whose scores are 75 or higher for passage retrieval.

Model	Heuristic	Retrieval	T-Test	Spearman's Correlation
Llama3	BLEU	Before	-0.2 (8.41e-01)	0.43 (1.86e-03)
Llama3	BLEU	After	3.75 (4.68e-04)	0.56 (2.20e-05)
Llama3	BERT	Before	-3.18 (2.58e-03)	0.07 (6.27e-01)
Llama3	BERT	After	-3.76 (4.48e-04)	0.01 (9.35e-01)
Gemma2	BLEU	Before	1.6 (1.16e-01)	0.29 (3.96e-02)
Gemma2	BLEU	After	5.09 (5.72e-06)	0.64 (5.83e-07)
Gemma2	BERT	Before	-5.28 (2.98e-06)	0.21 (1.34e-01)
Gemma2	BERT	After	-2.3 (2.57e-02)	-0.03 (8.20e-01)

Figure 5.18: T-Test and Spearman's Correlation (with Corresponding p-values) of BLEU and BERTScore for P_1 Using Mixtral7b Base Retrieval and Embedded Chunk Retrieval on Llama3 and Gemma2 only on Characters with all Selected Embedded Chunks having a Score of 75 or higher

In conclusion, even with effective passage embedding retrieval, the BLEUScore still worsens, but we observe a significant improvement in BERTScore across both models. Gemma2 does not benefit as much from passage embedding retrieval as Llama3, but it still delivers a much better BERTScore than Mixtral7b out of the box. The likely reason for the BLEUScore remaining similar without passage retrieval and then worsening in the passage embedding experiment with both models is the difference in context sizes: Mixtral7b has a context size of 32k tokens, whereas Gemma2 and Llama3 only have 8196 tokens. This can notably impact the results even with the same prompt length. A fair comparison between the models in any case is quite challenging due to their differing performances on various prompts. Their distinct architectures and vastly different training datasets make it nearly impossible to craft a query that elicits same behavior across all models. Finally, I provided the same scatterplots for Llamas BLEU- and BERTScores with passage embedding retrieval as in 5.8, but with color-coding for each franchise. The results demonstrate how different franchises improve at varying rates.

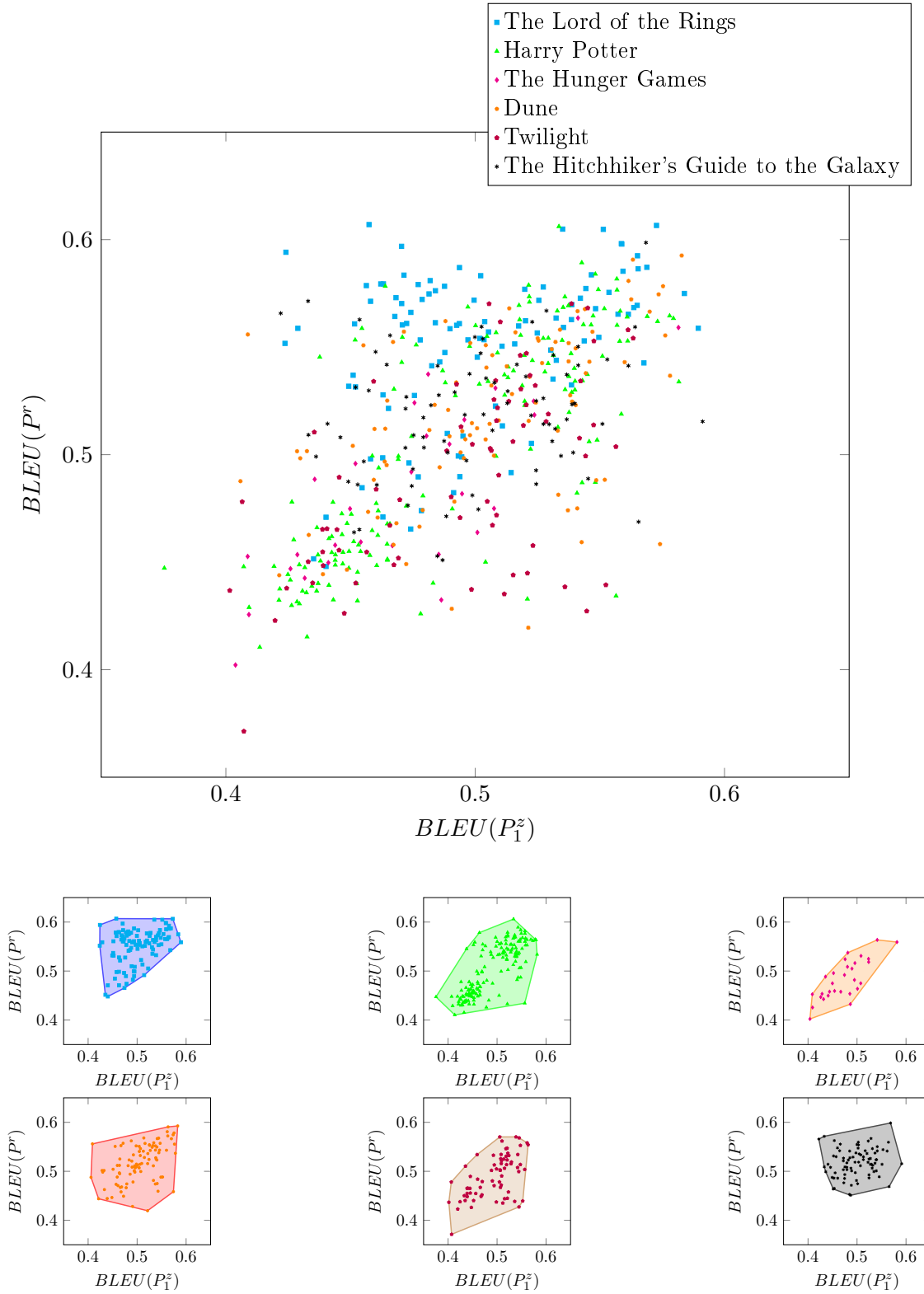


Figure 5.19: Scatterplot of BLEUScores of character descriptions generated with Llama3 on P_1 without (P^z) and with passage embedding retrieval (P^r) plotted against each other, color-coded and dissected into franchises and surrounded with convex hulls.

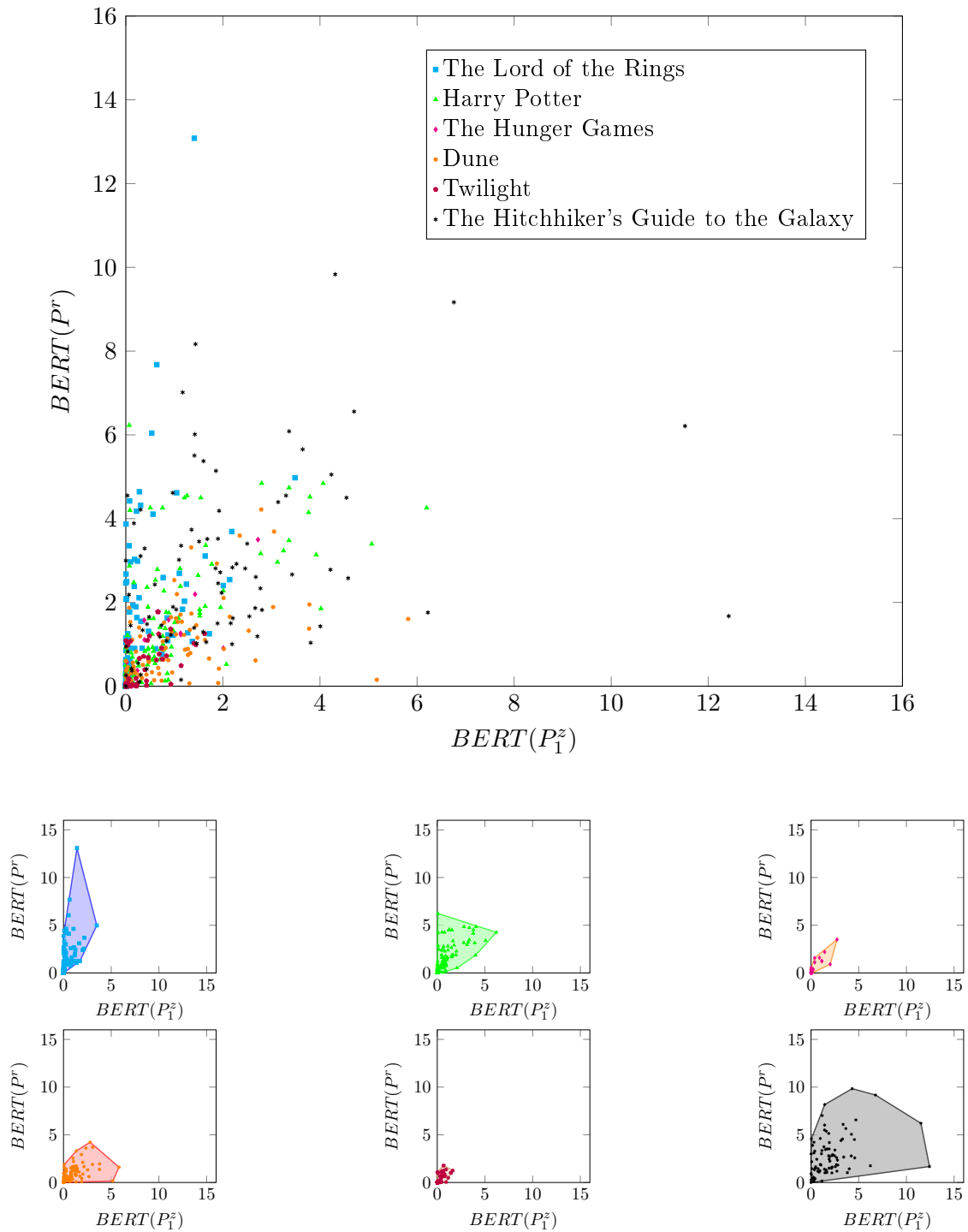


Figure 5.20: Scatterplot of BERTScores of character descriptions generated with Llama3 on P_1 without (P^z) and with passage embedding retrieval (P^r) plotted against each other, color-coded and dissected into franchises and surrounded with convex hulls.

6 Conclusion

6.1 Summary

Although different prompt wordings have only a slight impact on both BLEU and BERTScore, P_1 appears to be the clear winner. My experiments have clearly shown that passage retrieval positively impacts the generation of character descriptions. Passage embedding retrieval also outperforms baseline retrieval, but only with respect to characters with high chunk embedding scores, which have been shown to correlate with complete franchise datasets that do not miss any novels. The inconsistency of the dataset in particular missing novels for the corresponding franchise but also the inclusion of characters in fandom pages from similar works by the same author, different media like TV shows or older versions, has a measurable negative impact on the results. This likely affects passage embedding retrieval more severely than baseline retrieval because passage embedding retrieval still results in a fixed amount of passages that do not contain the mentioned character, thereby confusing the LLM. In contrast, baseline retrieval simply does not include any sentences at all for that character, making the results quite similar to those without passage retrieval. Llama3 and Gemma2 produce very different results for the same prompt. Gemma2 is more consistent in BLEUScore, whereas Llama3 achieves better BERTScore results. Gemma2 uses markdown elements like bullet points and sections and adds information in the response that goes beyond the desired result and is hindering. The reason why passage embedding retrieval negatively impacts BLEUScore is likely due to Mixtral7b's 32k context size. The larger input size allows for more input passages, and therefore, Mixtral7b captures more of the fandom articles' thesaurus compared to the other models.

6.2 Future Work

One of the most immediate challenges I encountered during my thesis was the significant amount of time required to scrape individual PDFs. Each editor structures their book differently, with varying placements for page numbers, chapter numbers, and titles, making it difficult to devise an efficient extraction method that works universally. But having a large and clean dataset is crucial, as it forms the foundation for everything that follows. It therefore would be reasonable to expand the amount of literature and character descriptions and to further refine the already carefully filtered dataset.

Another area for improvement would be the data filtering process of the human-written fandom articles from fandom.com. These articles often contain a wealth of detail and can be

quite lengthy, which led me to truncate them to a specific paragraph. My initial approach was to cut off the end of the articles, but there may be more effective ways to compress these articles e.g. using LLMs to shorten the summaries (although this also comes with a new set of potential correctness concerns) or possibly use them in their entirety without truncation. Further improvements might be achievable by applying coreference resolution techniques ([Dob21; SHB21]) to identify all tokens that refer to the given entity. If it is possible to identify self-contained content scopes using coreference resolution and segmenting the content by highly self-referenced text passages, the language model can generate even better character profiles due to the additional relevant information. Unfortunately this is currently hampered by coreference resolution quality. Additionally, fine-tuning models specifically for the task of creating character profiles presents another opportunity to align the results more closely with the desired outcomes. Since I used distinct models for baseline retrieval and passage embedding retrieval but did not apply both retrieval methods to a single model, we still need to conduct an experiment using both methods on the same model to obtain model-independent results.

For Evaluation, I primarily used BLEU and BERTScore metrics, although there are plenty more metrics and methods, some of which may be more discriminative with regard to the properties we look for in a good summary. As explored in [Yua+24], generating questions about characters and attempting to answer them using the generated summaries with a large language model (LLM), or even having an LLM evaluate the summaries on a scale, could provide deeper insights. It is always quite hard to establish a trustworthy relationship between the models output and the users needs, since we have no control on what pretrained- and passage retrieval data the LLM is relying on and which parts of it it is combining for its actual response. But there are still some ways to reject the obvious. Since language models are typically trained on extensive data, they might already contain information about certain books. To test this, we can compare queries that include key sentences to those that omit them. If the model produces the same output despite the missing key information, it suggests prior training on that data. Additionally, using books released after the model's training period ensures no pre-existing knowledge about the characters at all. Although I analyzed the results for each different book at the end of my experiments, a comparison between results for content that the LLM may have prior training knowledge of and newly released novels could put my findings even more into perspective. As LLMs continue to improve and increase their maximum input size to the point where handling the entirety of a book becomes feasible, the focus may shift to enhancing the quality and relevance of the information the LLMs process.

6.3 Acknowledgments

I would like to extend my heartfelt gratitude to all those who have supported me, knowingly or unknowingly, throughout the completion of this master's thesis. First and foremost, I am deeply grateful to my advisor, Hans Ole Hatzel, for his invaluable guidance and insights into NLP, which have been instrumental in shaping this research. I also wish to express my profound appreciation to my girlfriend, family and friends. Their unwavering love, support, and understanding throughout this journey have provided me with the motivation and inspiration needed to complete this work. Finally, I am thankful to everyone, both named and unnamed, who has directly or indirectly contributed to the success of this thesis.

7 Appendix

Codebase

The codebase repository, accessible at https://gitlab.rrz.uni-hamburg.de/BAR9094/nlp_thesis_scripts, contains all the programming scripts, both used on my personal computer and the remote server from the LT group, developed and utilized throughout this thesis.

LaTeX

The LaTeX files used for the compilation of this thesis are stored in the repository at https://gitlab.rrz.uni-hamburg.de/BAR9094/nlp_thesis_latex.

Bibliography

- [Ama24] Amazon Web Services. *What is Retrieval-Augmented Generation?* Accessed: 2024-08-14. 2024. URL: <https://aws.amazon.com/de/what-is/retrieval-augmented-generation/>.
- [Bra+21] Faeze Brahman et al. ““Let Your Characters Tell Their Story”: A Dataset for Character-Centric Narrative Understanding”. In: *Findings of the Association for Computational Linguistics: EMNLP 2021*. Ed. by Marie-Francine Moens et al. Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 1734–1752. DOI: 10.18653/v1/2021.findings-emnlp.150. URL: <https://aclanthology.org/2021.findings-emnlp.150>.
- [Che+24] Jeffrey Cheng et al. *Dated Data: Tracing Knowledge Cutoffs in Large Language Models*. 2024. arXiv: 2403.12958 [cs.CL]. URL: <https://arxiv.org/abs/2403.12958>.
- [Dob21] Vladimir Dobrovolskii. “Word-Level Coreference Resolution”. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Ed. by Marie-Francine Moens et al. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 7670–7675. DOI: 10.18653/v1/2021.emnlp-main.605. URL: <https://aclanthology.org/2021.emnlp-main.605>.
- [Dos21] Ketan Doshi. *Foundations of NLP Explained — Bleu Score and WER Metrics*. <https://towardsdatascience.com/foundations-of-nlp-explained-bleu-score-and-wer-metrics-1a5ba06d812b>. 2021.
- [Dub+24] Abhimanyu Dubey et al. *The Llama 3 Herd of Models*. 2024. arXiv: 2407.21783 [cs.AI]. URL: <https://arxiv.org/abs/2407.21783>.
- [Fac24] Hugging Face. *Tokenizer Summary*. Accessed: 2024-08-14. 2024. URL: https://huggingface.co/docs/transformers/en/tokenizer_summary.
- [Hug] Hugging Face. *Mixtral-Instruct-8x7b-Quantized-GGUF*. <https://huggingface.co/ikawrakow/mixtral-instruct-8x7b-quantized-gguf>. Accessed: 2024-07-19.
- [Int] Intfloat. *E5-Mistral-7b-Instruct*. <https://huggingface.co/intfloat/e5-mistral-7b-instruct>. Accessed: 2024-08-11.

-
- [NFK02] Joel Larocca Neto, Alex A. Freitas, and Celso A. A. Kaestner. “Automatic Text Summarization Using a Machine Learning Approach”. In: *Advances in Artificial Intelligence*. Ed. by Guilherme Bittencourt and Geber L. Ramalho. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 205–215. ISBN: 978-3-540-36127-5.
- [Pro23] Project Gutenberg. *Project Gutenberg*. Accessed: 2024-08-06. 2023. URL: <https://www.gutenberg.org>.
- [Ras24] Sebastian Raschka. “Understanding Encoder and Decoder Architectures in Deep Learning”. In: *Sebastian Raschka’s Magazine* (2024). Accessed: 2024-08-14. URL: <https://magazine.sebastianraschka.com/p/understanding-encoder-and-decoder>.
- [SHB21] Fynn Schröder, Hans Ole Hatzel, and Chris Biemann. “Neural End-to-end Coreference Resolution for German in Different Domains”. In: *Proceedings of the 17th Conference on Natural Language Processing (KONVENS 2021)*. Ed. by Kilian Evang et al. Düsseldorf, Germany: KONVENS 2021 Organizers, Sept. 2021, pp. 170–181. URL: <https://aclanthology.org/2021.konvens-1.15>.
- [suv24] suvratarora06. *NLP - BLEU Score for Evaluating Neural Machine Translation – Python*. <https://www.geeksforgeeks.org/nlp-bleu-score-for-evaluating-neural-machine-translation-python/>. 2024.
- [THB24] † Talika Gupta, ‡ HansOleHatzel, and Christian Biemann. “Coreference in Long Documents using Hierarchical Entity Merging”. In: *LATECHCLFL*. 2024. URL: <https://api.semanticscholar.org/CorpusID:267336474>.
- [Tur] Turing. *Fine-Tuning LLMs: Overview, Methods, and Best Practices*. Accessed: 2024-07-19.
- [Vas+17] Ashish Vaswani et al. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.
- [W3C08] W3C. *SPARQL Query Language for RDF*. Accessed: 2024-08-14. 2008. URL: <https://www.w3.org/TR/rdf-sparql-query/>.
- [Wik24] Wikidata. *Wikidata: SPARQL tutorial*. Accessed: 2024-08-14. 2024. URL: https://www.wikidata.org/wiki/Wikidata:SPARQL_tutorial.
- [Yua+24] Xinfeng Yuan et al. *Evaluating Character Understanding of Large Language Models via Character Profiling from Fictional Works*. 2024. arXiv: 2404.12726 [cs.CL]. URL: <https://arxiv.org/abs/2404.12726>.
- [Zha+20] Tianyi Zhang* et al. “BERTScore: Evaluating Text Generation with BERT”. In: *International Conference on Learning Representations*. 2020. URL: <https://openreview.net/forum?id=SkeHuCVFDr>.
-

List of Figures

2.1	Transformer Architecture as presented in the original paper [Vas+17]. This diagram illustrates the various layers of the architecture within the Encoder and Decoder, demonstrating the flow of data through each component. . . .	4
4.1	Number of Fandom Articles included in our Dataset for each Work	18
4.2	List of Individual Novels per Franchise Utilized for Passage Retrieval in the Experiments	19
4.3	SPARQL query to retrieve selected items, fandoms, fandom statements, and character names filtered by the "lotr" fandom from wikidata.	20
4.4	Sample Fandom Article for the Character "Gollum" from "The Lord of the Rings" Fandom Site (https://lotr.fandom.com/wiki/Gollum)	21
5.1	Structures of All Prompts Used for Zero-Shotting and Passage Retrieval in the Baseline Experiment	24
5.2	Scatterplot of BLEUScores of character descriptions generated with Mixtral7b without (P^z) and with baseline passage retrieval (P^r) plotted against each other	26
5.3	Scatterplot of BERTScores of character descriptions generated with Mixtral7b without (P^z) and with baseline passage retrieval (P^r) plotted against each other	27
5.4	Sorted differences between BLEUScores of character descriptions generated with Mixtral7b without (P^z) and with baseline passage retrieval (P^r) for every prompt (P_1, P_2, P_3, P_4).	28
5.5	Sorted differences between BERTScores of character descriptions generated with Mixtral7b without (P^z) and with baseline passage retrieval (P^r) for every prompt (P_1, P_2, P_3, P_4).	29
5.6	Boxplots of BLEU- and BERTScores of character descriptions generated with Mixtral7b for every prompt (P_1, P_2, P_3, P_4) with and without baseline passage retrieval.	30
5.7	T-Test and Spearman's correlation (with Corresponding p-values) of BLEU- and BERTScores of character descriptions generated with Mixtral7b for every prompt (P_1, P_2, P_3, P_4) with and without baseline passage retrieval. . . .	31

5.8	Scatterplot of BLEUScores of character descriptions generated with Llama3 and Gemma2 on P_1 without (P^z) and with passage embedding retrieval (P^r) plotted against each other.	34
5.9	Scatterplot of BERTScore of character descriptions generated with Llama3 and Gemma2 on P_1 without (P^z) and with passage embedding retrieval (P^r) plotted against each other.	34
5.10	T-Test and Spearman's Correlation (with Corresponding p-values) after prompting Llama3 and Gemma2 with P_1 with and without passage embedding retrieval	35
5.11	Sorted differences between BLEU- and BERTScores of character descriptions generated with Llama3 and Gemma2 of P_1 without (P_1^z) and with passage embedding retrieval (P_1^r).	35
5.12	Prompting result for the character "Nymphadora Tonks" from the Harry Potter franchise generated with P_1^r (passage embedding retrieval) in Gemma2.	36
5.13	Sorted Differences of BLEU- and BERTScores for P_1 Between Baseline Retrieval on Mixtral7b and Passage Embedding Retrieval on Llama3 Before and After Retrieval	38
5.14	Sorted Differences of BLEU- and BERTScores for P_1 Between Baseline Retrieval on Mixtral7b and Passage Embedding Retrieval on Gemma2 Before and After Retrieval	39
5.15	T-Test and Spearman's Correlation (with Corresponding p-values) of BLEU- and BERTScore for P_1 Using Base Retrieval and Selected Embedded Chunk Retrieval on Llama3 and Gemma2	39
5.16	Distribution of the embedding scores of all passages used for passage retrieval computed with e5-mistral.	40
5.17	Distribution of embedding similarity scores (in percentages) for all passages used for retrieval, categorized by fandom and score interval, calculated using e5-mistral.	41
5.18	T-Test and Spearman's Correlation (with Corresponding p-values) of BLEU and BERTScore for P_1 Using Mixtral7b Base Retrieval and Embedded Chunk Retrieval on Llama3 and Gemma2 only on Characters with all Selected Embedded Chunks having a Score of 75 or higher	42
5.19	Scatterplot of BLEUScores of character descriptions generated with Llama3 on P_1 without (P^z) and with passage embedding retrieval (P^r) plotted against each other, color-coded and dissected into franchises and surrounded with convex hulls.	43
5.20	Scatterplot of BERTScores of character descriptions generated with Llama3 on P_1 without (P^z) and with passage embedding retrieval (P^r) plotted against each other, color-coded and dissected into franchises and surrounded with convex hulls.	44

Affidavit

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Masterstudiengang Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe.

Ich stimme der Einstellung der Arbeit in die Bibliothek des Fachbereichs Informatik zu.

Hamburg, den _____ Signature: _____