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

Inhaltsverzeichnis

1 Introduction

Embarking on the literary journey of a captivating book, one often finds themselves 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 improved, meticulous and detailed personal descriptions for each character.

An important aspect of this study involves compiling existing literature and human-written characterizations of its characters, allowing for a comparison between these established descriptions and the results generated by large language models (LLMs) when given various prompts created using different techniques that extract information from the according literatur. The main focus will be on the methods for extracting and modifying information from literatur in order 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 running literature into different text formats. It is particularly interesting to observe how LLMs, such as LLaMA, respond to slight variations in query formulations. 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 entitiy from a text.

I intend to structure the thesis around the experiments and their results. I believe that segregating the experiments with formal sections would disrupt the central theme, making it harder for both me and the reader to follow. Therefore, I will provide a broad overview of the methods I might consider using in the methodology section and then each experiment will be discussed and evaluated individually.

2 Methodology

2.1 Embeddings/Token

2.2 The Transformer

The Transformer architecture is a powerful tool in natural language processing (NLP) tasks, particularly adept at sequence-to-sequence problems like machine translation. It relies on an encoder-decoder structure to process and generate textual data. The encoder processes the input sequence and generates a contextualized representation for each token. This encoded representation is passed to the decoder. The decoder starts generating the output sequence one token at a time. At each step, it uses masked self-attention to consider previously generated tokens and encoder-decoder attention to refer back to the encoded source sentence. Based on this combined information, the decoder predicts the next most likely token in the output sequence.

2.2.1 Encoder

The encoder takes an input sequence, like a sentence, and breaks it down into individual tokens (words or sub-words). Each token is mapped to an embedding vector, capturing its semantic meaning. The encoder then uses multiple encoder blocks to process the sequence. Each encoder block contains two sub-layers: Self-attention mechanism: This allows the model to understand the relationships between different words in the input sequence. It analyzes each token and assigns weights to other tokens based on their relevance. This creates a contextually rich representation for each token. Feed-forward neural network: This further refines the encoded representation by applying non-linear transformations.

2.2.2 Decoder

The decoder's role is to generate the output sequence based on the encoded representation from the encoder. It also uses multiple decoder blocks, similar to the encoder. Each decoder block has two additional sub-layers compared to the encoder block: Masked self-attention: Similar to the self-attention in the encoder, but it only attends to the previously generated tokens in the output sequence to avoid predicting future words based on information not yet generated. Encoder-decoder attention: This mechanism allows the decoder to attend to the encoded representation from the encoder, effectively looking back at the source sentence for context when generating the target sequence. Overall Process:

2.3 BERT

BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained language representation model introduced by Devlin et al. in 2019 (ref). Its based on the Transformer architecture from. . . 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. why?

At that time, it achieved state-of-the-art results on many common NLP tasks such as sentiment analysis, text prediction, or named entity recognition.

BERT introduces two pre-training objectives, the masked language model objective, and the next sentence prediction objective. a. This pre-training involves two main tasks:

Masked Language Modeling (MLM): Predicting masked words in a sentence, considering both left and right context. Next Sentence Prediction (NSP): Classifying whether two sentences follow each other logically.

After pre-training, BERT can be fine-tuned for specific tasks by adding a small layer on top of the pre-trained model. This fine-tuning is much faster and cheaper than pre-training the entire model from scratch.

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.

2.3.2 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 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. 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).
cosine similarity?

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 and is providing a balanced measure that considers both the model's ability to capture relevant information and its accuracy in predicting new text equally.

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

2.4 LLMs

2.5 BLEU-Score

2.6 Query generation

There are multiple approaches to consider when selecting the optimal prompts and information for a language model to generate high-quality summaries or characterizations. While rephrasing a task for a language model can influence the prompt's outcome to some degree, this study will primarily focus on selecting and curating the relevant literature data, which will be appended to the task of the prompt.

One method to obtain the necessary information is to filter the text 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 also additionally, this approach might include too much unnecessary information, especially for main characters, making it unsuitable for zero-shot character summary generation.

(https://huggingface.co/intfloat/e5-mistral-7b-instruct) expand... To improve upon this, text embeddings can be utilized. For instance, using a model like E5-Mistral-7B-Instruct, which has 32 layers and an embedding size of 4096, we can chunk the literature into sections of roughly equal length and embed each chunk. This allows us to identify chunks that satisfy a certain premise, such as describing a particular character more accurately than others.

Further improvements can be achieved by applying coreference resolution techniques ([SHB21], [Dob21]) to identify all tokens that refer to the given entity. This helps in gathering more sentences relevant to the characters context.

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.

Another approach to consider is fine-tuning a language model like LLAMA to enhance text summarization results.
The generated characterizations can be evaluated both qualitatively and quantitatively. To compare human-written characterizations with those generated by the model, we can measure recall and precision using metrics like ROUGE and BLEU, and employ BERTS-

core as a semantic evaluation metric.

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.

Existing human-written characterizations will serve as benchmarks for assessing the model's output in terms of style, content, structure, and level of detail.

i set up a ssh connection with the nlp-servers on the informatikum in order to be able to use their Llama models on their much more powerful GPUs and used HTTP-Requests in order to make prompts and get the results. For the weights of the model i used:

2.7 weights quantized uniserve3r llama.cpp

3 Related Work

Now first of all there already has been a decent amount of approaches for automatic text-summarization. One of the oldest and most cited papers belongs to Äutomatic Text Summarization Using a Machine Learning Approach"[NFK02]. It describes a summarization procedure based on trainable Machine Learning algorithms. Creating a Characterization is quite similar to making a Summarization of character related content but could also include deductions made from the behavior of that character. A recent Paper from 2021 [Bra+21] presents a dataset called LiSCU (Literary Summaries with Character Understanding) that aims to facilitate research in character-centric narrative understanding. They used techniques for 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. In contrast to their approach for Character Description Generation which required modeling long-range dependencies, I am using Retrieval-augmented generation (RAG), which is a technique to improve the quality of LLM-generated responses by grounding the model on external sources. LLMs are inconsistent in terms of producing same quality responses for each and every topic, since they knowledge is based on finite amount of information, that isn't 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. In this thesis I am primarily focusing on getting those important properties and behavior (key features) from the characters described in the literature to achieve better characterizations with grounded models that utilize this external information.

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,ör Älice's Adventures in Wonderlandäre rare cases where enough fandom exists to create accessible and reviewed content. In most other instances, it seems too risky to use open-source 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.

All of the books contained 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 before further processing, as they interfered with some of the techniques applied later.

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 chose 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 Encyclopediaänd Ëxpanded Dune,äs 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.

So 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 URLs by hand. 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 800 characters from eight books in total. The results are linked in the appendix.

Book	**Amount**
Harry Potter	157
Dune	103
Twilight	72
Alice's Adventures in Wonderland	30
The Lord of the Rings	53
The Hitchhiker's Guide to the Galaxy	90
The Hunger Games	29
Total	534

Tabelle 4.1: dataset of characters and their descriptions

...

Project Gutenberg and the Tell Me Again! Dataset...

5 Experiments

All my experiments have been conducted partially on my own computer but also on a remote server from the LT group at the University of Hamburg. This was mainly due to accessing a better GPU like the RTX A1000 NVIDEA for prompting LLMS and creating embeddings.

5.1 Base 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 rely solely on its training data. "[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 from the book for the given character that might help the Llama model with its characterizations will be passed into the prompt.

Prompt	Instruction
δ_1	"[INST]Write a summary about the character {character} in the book {book}.[/INST]"
δ_1'	"[INST]Write a summary about the character {character} in the given text passages: \n {help}[/INST]"
δ_2	"[INST]Write a summary in the style of a fandom article about the character {character} in the book {book}.[/INST]"
δ_2'	"[INST]Write a summary in the style of a fandom article about the character {character} in the given text passages: \n {help}[/INST]"
δ_3	"[INST]Provide a concise overview of the character {character} from the book {book}.[/INST]"
δ_3'	"[INST]Provide a concise overview of the character {character} based on the following excerpts: \n {help}[/INST]"
δ_4	"[INST]rite sumary bout thee cara cter {character} of th book {book}.[/INST]"
δ_4'	"[INST]rite sumary bout thee cara cter {character} bsed th fllowing excerpts: \n {help}[/INST]"

As you can see, δ_2 is more specific, requesting the style of a fandom article, whereas δ_3 is less precise, asking only for an overview without specifying a particular format. The last prompt is similar to δ_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 this first experiment, I selected additional information from the book 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, because characters are more likely to be introduced in the first sentences where they appear in the book, I added an additional cutoff α. This cutoff represents the percentage of relevant sentences (with character name occurrences) to which every sentence with name occurence 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 containig the character and l be the maximum inputsize of the Llama query. We first definde 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

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

we can write R_{base} as

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

. I fed the prompts through the mixtral 7b model with quantized weights. Quantization is a method used to decrease the computational and memory demands of running inference by using low-precision data types, such as 8-bit integers (int8), instead of the standard 32-bit floating-point (float32). Using fewer bits reduces the memory storage needed for the model, theoretically lowers energy consumption, and speeds up operations like matrix multiplication through integer arithmetic. This technique also enables models to run on embedded devices, which may only support integer data types. They are different types and levels of quantization and i started with the smallest $Q2_K$ (https://huggingface.co/ikawrakow/mixtral-instruct-8x7b-quantized-gguf) weights and therfor quickest responses. For the evaluation, I used BLEUScore and BERTScore to compare the results from the prompts against manually written articles from fandom.com.

5.1.1 Results

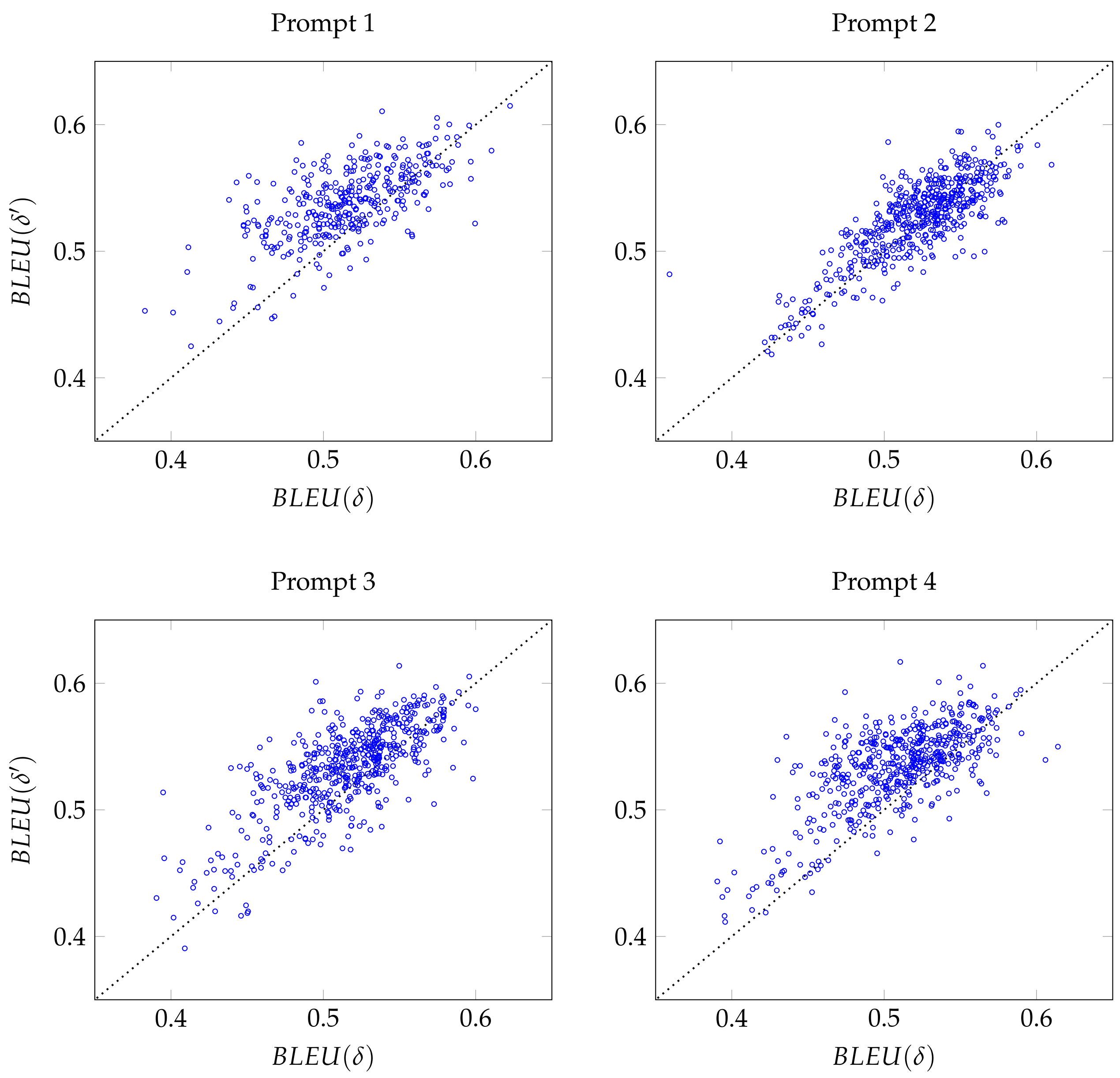


Abbildung 5.1: Blue-Metric of Zero-Shot Characterizations generated with Llama without (δ) and with passage retrieval (δ') from the literature

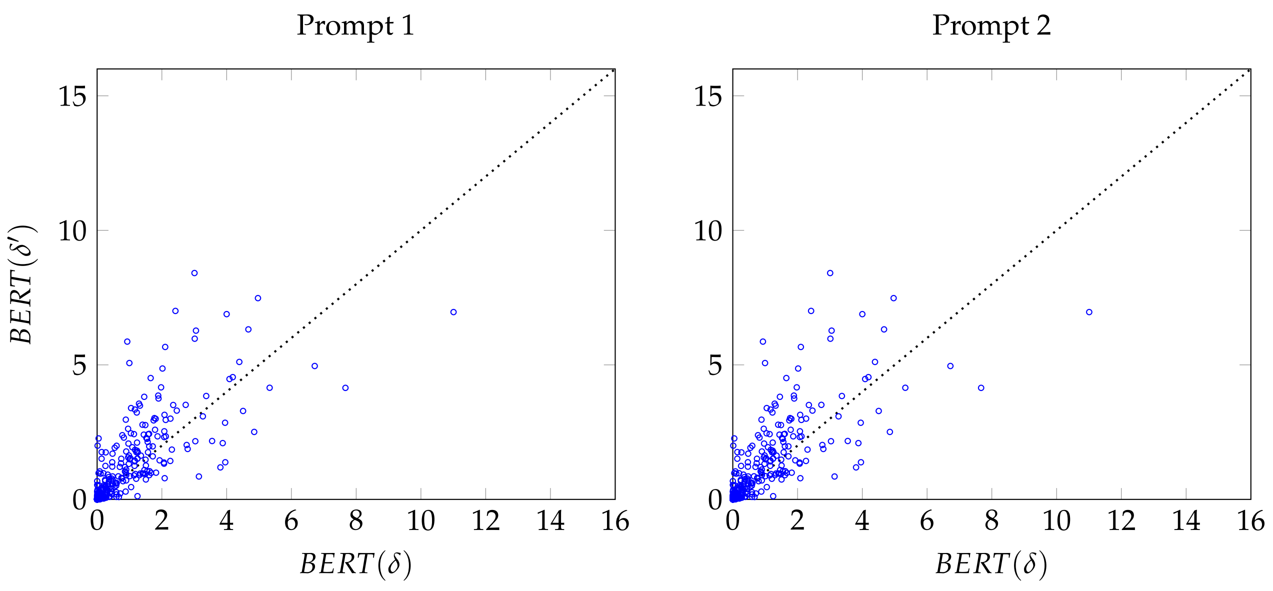
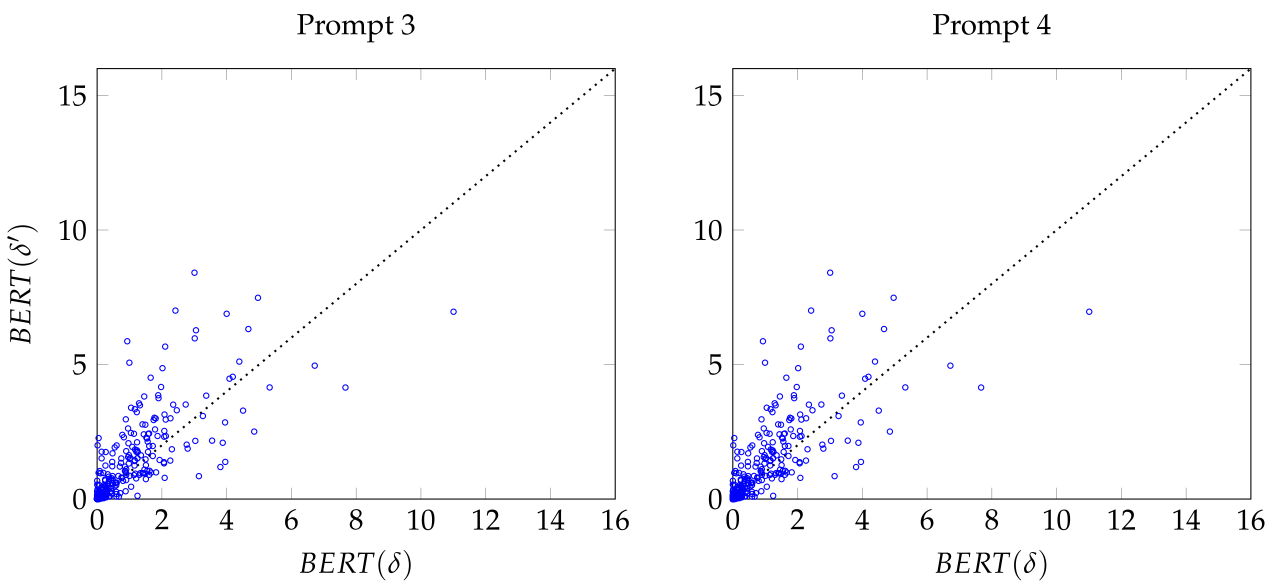
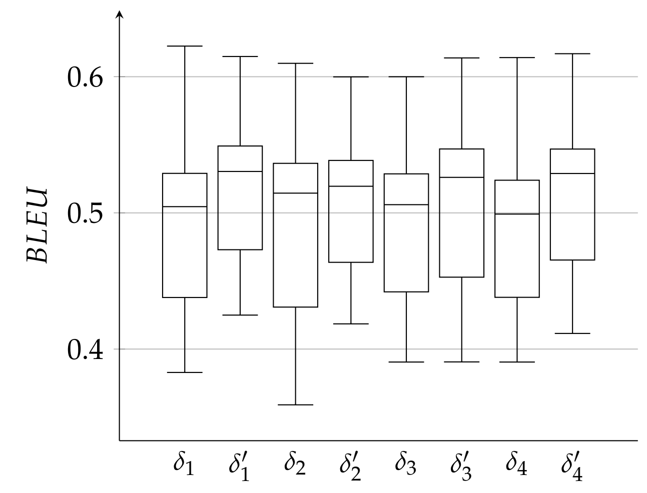



Abbildung 5.2: BERTScore of Zero-Shot Characterizations generated with Llama without (δ) and with passage retrieval (δ') from the literatur



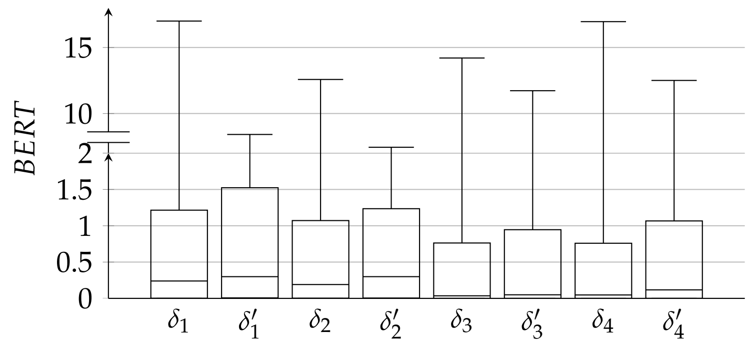


Abbildung 5.3: Boxplots of BLEU- and BERTScores for every prompt ($\delta_1, \delta'_1, \delta_2, ..., \delta'_4$)

5.1.2 Analysis

Obviously the method of passage retrieval used for this experiment isn't ideal, as regularly eliminating sentences could omit important context, also at this stage, the process of fetching fandom articles wasn't complete, resulting in a dataset with some duplicates and missing characterizations. Despite these limitations, the data is still sufficient to show two important aspects of the data. First, the results with passage retrieval are at least as good as, or already slightly better than, those without. Second, the results vary only slightly across the four different prompts. Befor we investigate that further lets have an more detailed look at the results.

As we can see, the BLEU scores of each prompt mostly improve after passage retrieval. Although the maximum values of δ_1 and δ_2 have decreased slightly in δ'_1 and δ'_2, the minimum values, Q1, and Q3 have significantly increased, as observed in the box plots. For both δ_3 and δ_4, every box plot quartile has improved.

For BERTScore, the improvement isn't quite as visible. In fact, the upper quartiles have a lower maximum after passage retrieval, but Q1-Q3 has improved slightly for every prompt. Consequently, the results are more compact. Some outliers close to the maximum in Q4 might score so high prior to passage retrieval due to Llama being trained on similar information to the fandom articles. Especially when generating summaries for main characters, Llama 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 characterization.

In summary, semantically, the results have only improved slightly and the different wordings in the prompts definately have an influence on the results average and variance (ref figure).

5.2 Selected Embedded chunks

We will now continue with prompt 1 from the base experiment since it had the highest results in both metrics. We will now improve on the method for passage retrieval.

6 Conclusion

6.1 summary

6.2 future work

6.3 note of thanks

Here I will describe future work that can be done to improve my work and talk about the advantages, disadvantages of the methods I used but also other problems that occured during the time (e.g. Iack of literatur) that I was able to obtain

Literatur

[Bra+21] Faeze Brahman u. a. ““Let Your Characters Tell Their Story”: A Dataset for Character-Centric Narrative Understanding”. In: *Findings of the Association for Computational Linguistics: EMNLP 2021*. Hrsg. von Marie-Francine Moens u. a. Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, S. 1734–1752. DOI: `10.18653/v1/2021.findings-emnlp.150`. URL: `https://aclanthology.org/2021.findings-emnlp.150`.

[Dob21] Vladimir Dobrovolskii. “Word-Level Coreference Resolution”. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Hrsg. von Marie-Francine Moens u. a. Online und Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, S. 7670–7675. DOI: `10.18653/v1/2021.emnlp-main.605`. URL: `https://aclanthology.org/2021.emnlp-main.605`.

[NFK02] Joel Larocca Neto, Alex A. Freitas und Celso A. A. Kaestner. “Automatic Text Summarization Using a Machine Learning Approach”. In: *Advances in Artificial Intelligence*. Hrsg. von Guilherme Bittencourt und Geber L. Ramalho. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, S. 205–215. ISBN: 978-3-540-36127-5.

[SHB21] Fynn Schröder, Hans Ole Hatzel und 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)*. Hrsg. von Kilian Evang u. a. Düsseldorf, Germany: KONVENS 2021 Organizers, Juni 2021, S. 170–181. URL: `https://aclanthology.org/2021.konvens-1.15`.

7 Appendix

In this section I will link my repositories containing all my scripts and data that I created and collected during my thesis.

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe angefertigt und mich anderer als der im beigefügten Verzeichnis angegebenen Hilfsmittel nicht bedient 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 und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.
Ich bin mit einer Einstellung in den Bestand der Bibliothek des Fachbereiches einverstanden.

Hamburg, den __________ Unterschrift: _________________________