Improving faithfulness in RAG systems

Luca Furtuna

YEAR OF STUDY: 2023-2024

Project Supervisors: Adel Bibi & Alasdair Paren

A DISSERTATION PROJECT REPORT SUBMITTED FOR THE DEGREE OF:

Masters of Science in Advanced Computer Science

Trinity 2024

Abstract

Large Language Models (LLMs) have demonstrated significant success in performing various natural language processing (NLP) tasks, leading to their increasing integration into critical fields such as medicine and law. However, one of their major limitations is the tendency to generate "hallucinations", or made-up answers. Retrieval-Augmented Generation (RAG) models were introduced to mitigate this issue, though they cannot fully solve it. This report presents a novel framework for RAGs aimed at reducing hallucinations, specifically in the context of open-domain question answering in law. The framework learns prompt embeddings through Prompt tuning, a Parameter-Efficient Fine-Tuning (PEFT) method. The trained soft prompts are, thus, prepended to an input question which enforces the RAG generator component to produce answers that closely reflect the retrieved documents, an essential feature in common law systems where factual accuracy is paramount. We achieve the development of the framework in Python, which manages to guide LLMs into extracting answers from the document in the input while often keeping the answer relevant and we further conduct experiments on baseline manually optimized prompts and the trained soft prompts to testify our claims.

Keywords Large language models, Prompt engineering, Retrieval-Augmented Generation, longest common subsequence, hallucination

Contents

Section 1

Introduction

Large Language Models (LLMs), such as OpenAI's *GPT* [1] or Google's *BERT* [2], are complex machine learning systems trained on enormous datasets containing diverse text from books, websites, and other written sources. These models are capable of NLP tasks such as generating text, answering questions, translation and summarization. Their applications span a wide range of industries, from automating customer service to enhancing education and providing conversational interfaces. In particular, LLMs have become indispensable tools in areas such as healthcare, legal, and financial sectors, demonstrating their high-impact potential in real-world applications.

Unfortunately, LLMs tend to often perform badly in knowledge-intensive tasks [3], which are NLP tasks that require synthesizing and memorizing facts from a knowledge source such as Wikipedia. One of the significant issues with LLMs is their tendency to "hallucinate" [4], that is, to generate information that sounds plausible but is factually incorrect or completely fabricated. This limitation arises because LLMs learn a probability distribution across language, which doesn't completely model the truth found in the real world. As such, verified facts may be unlikely to occur from a linguistic perspective, while some incorrect statements are likely to be generated. While this may be acceptable in casual conversation or creative writing, it poses a serious risk in fields such as medicine or law, where the consequences of misinformation can be costly.

To address this challenge, Retrieval-Augmented Generation (RAG) was designed. RAG integrates LLMs with external knowledge bases, retrieving relevant documents or data from trusted sources to provide contextually accurate information. RAG seeks to reduce the occurrence of hallucinations, improving the factual reliability of the generated output. This hybrid approach has been particularly valuable in applications where factual correctness is essential, such as research, legal documentation, or customer service involving technical information.

Despite the advantages, RAGs are not immune to producing hallucinations - instances

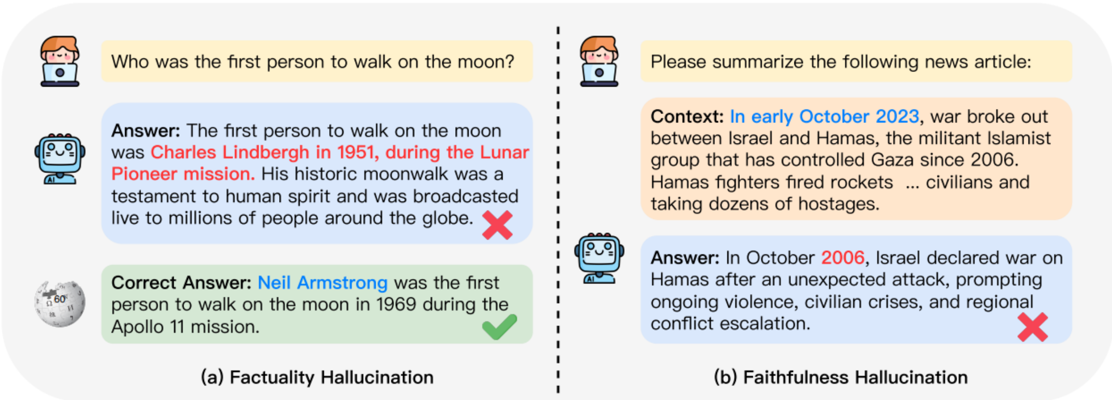


Figure 1.1: An example of a hallucination [5]

where the generated text includes incorrect or unfounded information or, *"unfaithful"* to the retrieved context [5]. Studying the effects and mitigations of hallucinations is an active area of research, especially in the context of law-related question answering [6]. In particular, a study showed that general-purpose LLMs hallucinate on law-related queries on average between 58% and 82% of the time [7].

We hypothesize that tuning an LLM for a more restrictive task such as outputting verbatim text, instead of just factually correct information, may prove to be an efficient method for combating hallucinations in RAGs. Furthermore, in certain scenarios, such as using LLMs to generate answers related to the law or policy of a company, it is important to have factually correct and relevant information generated as a response that resembles the context exactly. This holds especially in common law, where sometimes we want to form a decision based on the retrieved context which often contradicts the knowledge the LLM was trained on [6].

1.1 Objectives

In this project, our objective is to construct a framework that may be used within RAG systems that satisfies the following properties:

- The output of the RAG contains verbatim information extracted from the document to answer a question.
- The output of the RAG is relevant to the question.

We believe such a framework would help reduce hallucinations for general-purpose RAGs while also ensuring it uses the document to answer as opposed to knowledge stored in the model parameters, which would have significant applications in law.

We attempt to achieve this task by prompting the model, either through manual or automatic optimization, to extract as much relevant information as possible from a document provided in an input alongside a question in order to answer it.

1.2 Report structure

The remainder of in this report is structured as follows:

Section 2 introduces the terminology and necessary background to understand the subsequent sections. While we focus on presenting concise background material that is relevant to the project, we do often cite additional material, in case the reader is of interest. In Section 3, we rigorously formulate our task and present the methodology to achieve our objectives. We then go into detailing our experimental findings in Section 4, which support the capabilities of our framework. Lastly, in Section 5, we conclude our report, noting the contributions our work brings and promising future work.

Section 2

Background

In this section, we introduce all the background and terminology needed to understand the subsequent sections.

2.1 Transformers

We start by briefly going through the core architecture behind LLMs, namely *Transformer*. While it is a broad subject, we only outline the important concepts that we shall be using to complete our work. For further background on transformers, we refer the reader to [8], [9].

Transformer [9] is a deep learning architecture that was originally proposed as a competitive alternative to Recurrent Neural Networks (RRNs) and their limitations.

It consists of two main components - an **Encoder** and a **Decoder**, each playing its role in capturing long-range dependencies in sequential data.

2.1.1 Attention mechanism

The attention mechanism is the core idea behind the transformer architecture, which, intuitively, allows the model to comprehend contextual relationships within sequential data. Given the query matrix $\boldsymbol{Q} \in \mathbb{R}^{\mathsf{n} \times \mathsf{d}_k}$, the keys matrix $\boldsymbol{K} \in \mathbb{R}^{\mathsf{l} \times \mathsf{d}_k}$ and values matrix $\boldsymbol{V} \in \mathbb{R}^{\mathsf{l} \times \mathsf{d}_v}$, the scaled dot-product is defined as:

$$\textsc{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \textsc{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{\mathsf{d}_k}}\right)\boldsymbol{V} \tag{2.1}$$

where, in the context of transformers, n and l denote the number of tokens in the query set and keys/values set, respectively; d_v and d_k denote the dimension of values set and keys/queries set, respectively. We also define $\boldsymbol{A} = \frac{\boldsymbol{QK}^T}{\sqrt{d_k}}$ to be the unnormalized attention matrix. We also note that the matrix multiplication is divided by $\sqrt{d_k}$, in order to address the issue of vanishing gradient.

Transformers use **Multi-Head Attention**, which is defined as follows:

$$\text{MULTIHEAD}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = (\textit{head}_1 \oplus \cdots \oplus \textit{head}_h)\boldsymbol{W}^O \tag{2.2}$$

where $\textit{head}_i = \text{ATTENTION}(\boldsymbol{Q}\boldsymbol{W}_i^Q, \boldsymbol{K}\boldsymbol{W}_i^K, \boldsymbol{V}\boldsymbol{W}_i^V)$; $\boldsymbol{Q}_i$, $\boldsymbol{K}_i$ and $\boldsymbol{V}_i$ are linear layers; $\oplus$ denotes concatenation. In short, we independently compute h scaled-dot products of $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$, concatenate the outputs and apply a final linear layer $\boldsymbol{W}^O$.

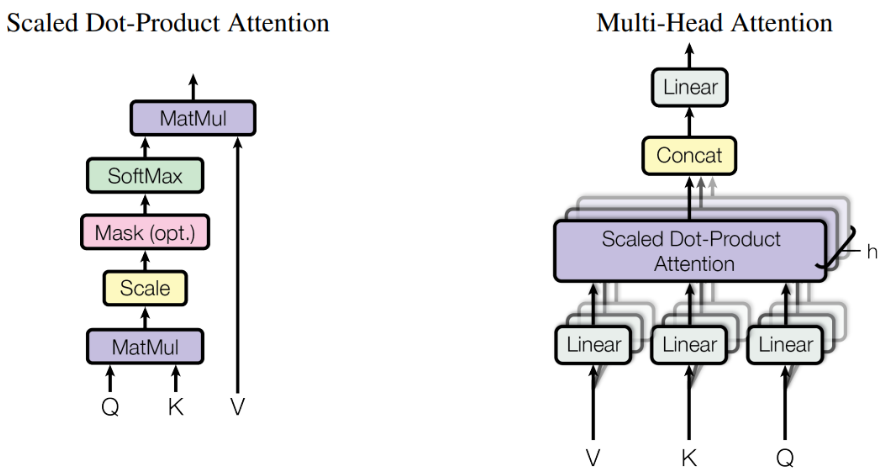


Figure 2.1: Attention Mechanism [9]

2.1.2 Embedding Layer

For a Transformer to be able to process text data, this data is first tokenized using a **tokenizer** and then fed into an **embedding layer** as input. The embedding layer can be thought of as a look-up table that maps discrete data such as tokens or words to a continuous vector space, usually $\mathbb{R}^{n}$ for some $n \in \mathbb{N}$.

In terms of implementation, an embedding layer is a matrix in $\mathbb{R}^{v \times n}$, where v is the size of the tokenizer's vocabulary and n is the **embedding size** or **hidden dimension**. The i-th row in the matrix is the vector representation of the i-th token and they are learned during the training process of the transformer.

We also note that the outputs of all sublayers (which are part of encoder/decoder blocks - or layers) in a Transformer are called the **hidden states** and they usually have the same hidden dimension, which is a hyperparameter. These matrices hold, for each token in the input/output sequence, its current vector representation.

Once the input/output has been projected onto a real space, positional encodings - which are vectors of embedding size that hold information about the position of each token in the input - are added to the input embeddings and output embeddings. The purpose of these vectors is to break the invariance to token permutation, which is an undesirable property since the order of words in a sentence matters in common NLP tasks.

Note We shall refer to the embeddings of a full or part of an input prompt $\boldsymbol{s}$ as $\hat{\boldsymbol{s}}$ and to the embeddings of a batch of inputs $\boldsymbol{B}$ as $\hat{\boldsymbol{B}}$.

2.1.3 Encoder

The vector representations are subsequently fed into the Encoder, a component that may contain multiple identical encoder blocks (which contain multiple sublayers) stacked on top of each other. The purpose of the component is to map the continuous representations of the input sequence to another representation that holds attention information.

Each block contains a multi-headed attention module as a sublayer, along with a fully connected neural network (*FFN*) used to further extract important features from the learned attention information. The *FFN* is applied individually to each vector output of the attention mechanism:

$$\text{FNN}(\boldsymbol{H}) = \text{ReLU}(\boldsymbol{H}\boldsymbol{W}_1 + \boldsymbol{b}_1)\boldsymbol{W}_2 + \boldsymbol{b}_2$$

where $\boldsymbol{H}$ is some hidden state and $\boldsymbol{W}_1$, $\boldsymbol{W}_2$, $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ are trainable parameters.

The Residual connections [10] and layer normalization [11] modules are further used to diminish the issues of vanishing or exploding gradients and stabilize the training process.

The attention mechanism module in an encoder block is particularly called a **Self-Attention** module, where $\boldsymbol{K} = \boldsymbol{Q} = \boldsymbol{V} = \boldsymbol{O}$ in Def. 2.2, where $\boldsymbol{O}$ is a hidden state. This way, each token in the input sequence learns which tokens from the input to attend to.

2.1.4 Decoder

Much like the encoder, the decoder is a component consisting of multiple blocks, whose role is to generate a sequence based on the input. The decoder outputs a new token at each inference given the input and the generated tokens before it. The overall sequence is said to be completely generated once the end-of-sequence token *<eos>* is generated or the maximum number of new tokens is reached. The main difference to the encoder module comes from the role of the two attention mechanism modules it consists of.

The first attention module, called the **Masked Multi-Head Attention** module, works similarly to the Self-Attention mechanism in the encoder module, with the restriction that each token can only attend to tokens that were generated before it. Thus, before applying SOFTMAX to the unnormalized attention matrix $\boldsymbol{A}$, we mask it by making sure

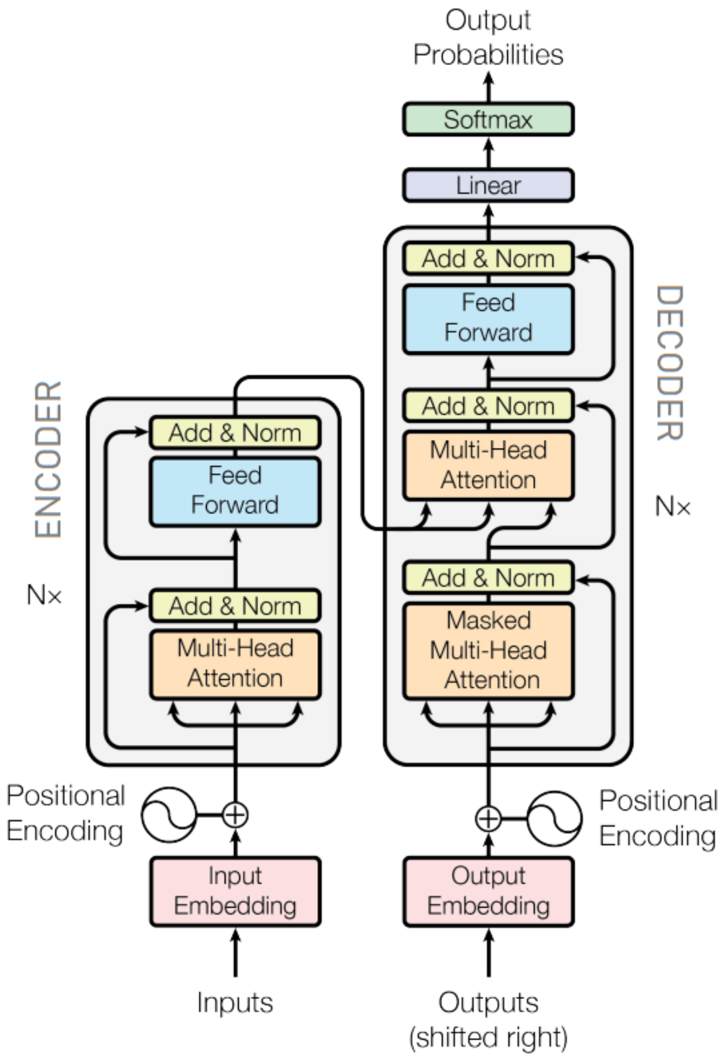


Figure 2.2: Transformer Architecture [9]

$\boldsymbol{A}_{ij} = -\infty$ if i < j.

In the second attention module, often referred to as **Cross-Attention**, we have $\boldsymbol{K} = \boldsymbol{V} = \boldsymbol{O}_E$ and $\boldsymbol{Q} = \boldsymbol{O}_D$ in Def. 2.2, where $\boldsymbol{O}_E$ denotes the outputs of the encoder module and $\boldsymbol{O}_D$ denotes the outputs of the previous masked-attention module. Intuitively, this attention mechanism learns to which tokens in the input the token currently being generated needs to attend.

The output of the decoder module is then fed into the final component which consists of a linear layer with a SOFTMAX layer applied to it. This module acts as a classifier which predicts the next token by computing, for each token in the output sequence - i.e. the currently generated sequence of tokens - the most likely token next to it. In the context of language models - see Sec. - this component is also called the **Language Model Head** (LM Head). The linear layer computes a matrix in $\mathbb{R}^{s \times v}$, where s is the size of the output sequence and v is, again, the vocabulary size. The entries of this matrix are also commonly referred to as **logits**. The SOFTMAX layer computes, for each token in the vocabulary, the probability that this is the token is the next token to be generated.

2.2 Causal Language Models

Transformer was initially introduced as a Sequence-to-sequence (Seq2Seq) model [12] for the machine translation NLP task. Nowadays, *state-of-the-art* language models which solve various NLP tasks are Transformer-based models.

Depending on the task, these models retain either one or both of the main components of the Transformer. For classification tasks such as sentiment analysis, encoder-only language models such as *BERT* [2] or *RoBERTa* [13] are favored; for sequence-to-sequence tasks such as machine translation and text summarization, encoder-decoder models such as *T5* [14] or *BART* [15] are typically used; finally, for text generation and question answering, decoder-only language models such as the *GPT* [1], [16], *LLama* [17], [18] or *Gemma* [19] series are the best performing among all the 3 variations. Since the task of question answering is the main focus of this project, we are mainly concerned with decoder-only models, often also called **Causal LMs** or **autoregressive models**.

Decoder-only Large Language Models (LLMs) are typically trained for text-generation, where the user input is a sequence and the model attempts to generate the rest of the sequence in an autoregressive manner, a process which is referred to as **pre-training** language models [20]. While pre-trained models learn general language representations [1], [2], [17] and to generate text, they may not perform optimally when applied to specific tasks like question answering, or sentiment analysis [21], [22]. Usually, such models are **fine-tuned** to follow instructions and align with user intent across a wide range of tasks, a process that involves reinforcement learning from human feedback (RLHF) or similar methods. Such models go by the name of **instruct models** [23] and they excel in tasks such as question answering, where the user inputs a question and the model continues the sequence by generating an answer

Note In this work, whenever we mention question answering (QA) as a task, we assume "open-domain" QA, where the model answers questions with context given in the input as supporting documents. In contrast, in the "closed-book" QA task, the model does not have access to any relevant context and solely relies in the knowledge encoded in its parameters to answer questions [24].

2.2.1 Architecture

The decoder-only Transformer has an architecture similar to the vanilla Transformer, except that each decoder block no longer has a Cross-Attention Mechanism, since we no longer make use of any outputs coming from an encoder.

A complete response is generated by an LLM by sampling the model in an autoregressive manner - where a token is predicted based on the tokens generated before it - up to the d_{out}-th token or until an end-of-sequence token *<eos>* is reached. There are multiple decoding methods, such as *Greedy Search*, *Beam Search* [26] or *Top-k sampling* [27]. In this project, we assume the foremost method, which selects the token with the highest probability at each decoding step. For more information on decoding methods, we refer the reader to [28].

Formally, we define a decoder-only LM $\mathcal{M}$ as $\mathcal{M}\colon \mathbb{V}^* \to \mathbb{V}$ - where $\mathbb{V}$ denotes the vocabulary associated with the LLM's tokenizer and $\mathbb{V}^*$ the set of strings constructed

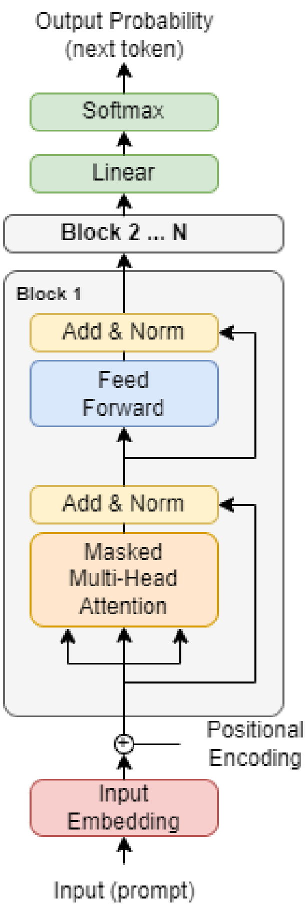


Figure 2.3: A decoder-only Transformer [25]

from the vocabulary - which outputs the next token with the highest probability given an input prompt s. Then, a complete response $\boldsymbol{y}$ to s (or as we should call it from this point forward, model **output**) is given by:

$$\boldsymbol{y} = f_{\mathcal{M}}(\boldsymbol{s}) \triangleq y_{d_{out}} \oplus \cdots \oplus y_1, \text{where } \; y_i = \mathcal{M}\left(\boldsymbol{s} \oplus \boldsymbol{y}_{[:i-1]}\right) \;\; \forall i \in \{1, \ldots, d_{out}\}$$

For brevity, we also define $f_{\mathcal{M}} \colon \mathbb{V}^* \to \mathbb{V}^*$ to be a function that maps an input prompt s to a complete response generated by the LLM.

2.2.2 Training

An autoregressive LM is typically pre-trained on a dataset $\mathcal{D}$ where each datapoint is a text $s \in \mathcal{D}$ whose ground-truth is itself shifted to the left by a position.

Given batch $\boldsymbol{B} = \{\boldsymbol{s}_i\}_{1 \leq i \leq N}$ of input prompts from dataset $\mathcal{D}$, probability distribution $\mathcal{P}\left(\cdot|\boldsymbol{s}_{i[:j-1]}\right)$ of the next predicted word at position j in the input $\forall j \in \{1, \ldots, \|\boldsymbol{s}_i\|\}$, we define the *Cross-Entropy* loss function as:

$$\mathcal{L}_{CE}(\Phi) = \frac{1}{N}\sum_{i=1}^{N}\left(-\frac{1}{\|\boldsymbol{s}_i\|}\sum_{j=1}^{\|\boldsymbol{s}_i\|}\log\left(\mathcal{P}\left(\boldsymbol{s}_{i[j+1]}|\boldsymbol{s}_{i[:j]}\right)\right)\right) \tag{2.3}$$

where the log($\cdot$) function is applied entry-wise and Φ are the model parameters.

Thus, in one training iteration, $\mathcal{L}_{CE}(\Phi)$ is computed and then backpropagation [29] is performed in order to update Φ.

2.3 Prompt engineering

2.3.1 Prompting

Prompting is the action of augmenting the user input with additional instructions manually designed to guide the LM into performing well for different tasks. Prompts typically include a description of the target task and several relevant examples. In contrast to fine-tuning, where the pre-trained model is being fully trained for a specific task which is memory-consuming due to the need of gradients to be stored in memory, prompting doesn't require any training - i.e. the model parameters are **frozen**, or not trainable. While manually designed prompts have been proven to be effective in altering the behavior of pre-trained models such as *GPT3* [1], it suffers from several drawbacks, such as the fixed size of **context window** (i.e. the maximum input length acceptable by an LM) of the LM which will often not be enough for the prompts to fit in. Efforts have been made to automatically optimize for the best prompt [30], [31], yet their main drawbacks are their intractable complexity intrinsic to optimizing over the discrete space of tokens and their often suboptimal performance compared to fine-tuning, which generally ensures good performance.

2.3.2 Soft prompt-based Fine-tuning

Unlike **hard prompting**, where we search for prompts to be augmented to the LM input over the discrete space of tokens $\mathbb{V}$, **soft prompting** optimizes over the continuous space representing the tokens that are projected onto by the embedding layer of the model. These continuous prompts are trainable and typically inserted into the embedded input or hidden states. Unlike hard prompts, soft prompts exhibit more flexibility and adaptability during training. Soft-prompting and other *Parameter-Efficient Fine-Tuning* (PEFT) methods [32]–[34] focus on reducing the memory and time expenses associated with fine-tuning pretrained models, as these methods freeze the model's parameters and train a small set of soft prompt vectors. Soft-prompt tuning has been proven to match fine-tuning performance for various tasks such as text summarization and question answering [32], [34].

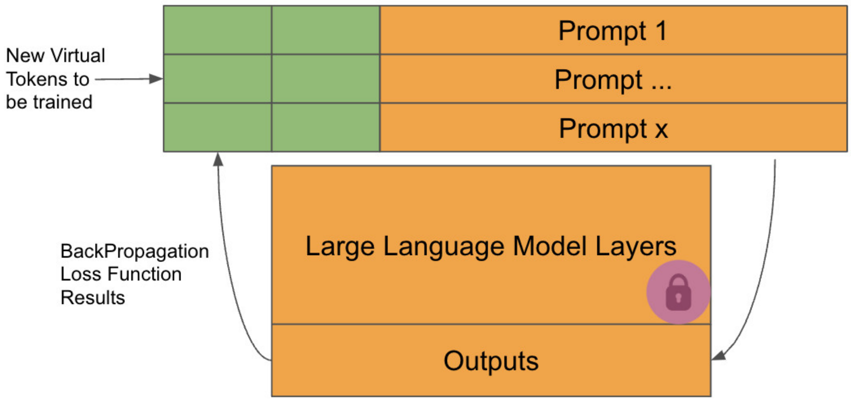


Figure 2.4: Prompt Tuning Architecture [35]

In particular, Prompt Tuning [32] prepends to an embedded input $\hat{s}$ of size k, n learnable embeddings $\hat{p} \in \mathbb{R}^{n \times e}$, where e is the token embedding dimension, resulting in the embedded full input:

$$\hat{s^*} = \hat{p} \oplus \hat{s} = [\hat{p}, \hat{s}] \in \mathbb{R}^{(n+k) \times e}$$

During training, only the prompt embeddings $\hat{p}$ are updated during gradient descent, while the model's parameters are frozen. That makes a total of $n \cdot e$ parameters, which is a substantially smaller number of parameters that need to be trained as opposed to the billions of pre-trained model parameters that need to be updated when fine-tuning it for a specific task.

In this project, we will be using Prompt tuning as a method for automating prompt searching. For more information on PEFT methods, we refer the reader to [36], [37].

2.4 Retrieval-Augemented Generation

RAG (Retrieval-Augmented Generation) is a model architecture introduced in the context of Long-Form Question Answering tasks [38], an emerging research area within question answering, that requires systems to generate paragraph-length answers to complex questions with the help of retrieved documents that are used as evidence, particularly in the context of large language models. RAG integrates a retriever component with a generative component.

The retriever part is responsible for efficiently finding relevant passages/documents from an external large corpus of text given a query. Typically, the corpus initially goes through an *indexing* process, through which the text is segmented into smaller chunks that can fit in the context window of an LLM. These chunks are then encoded into continuous vector representations using some embedding model, not necessarily the embedding layer of the LLM which is integrated within the RAG system. Thus, this enables the retrieval systems to perform simple algorithms like returning the top K chunks with the greatest similarity to the query. There are various other retrieval algorithms which, for the sake of brevity, we won't elaborate, since in our work we assume factually correct and relevant retrieved context when designing our models. For further reading into RAGs and retrieval techniques, we refer the reader to [39], [40]

The generative component then generates the response based on the retrieved

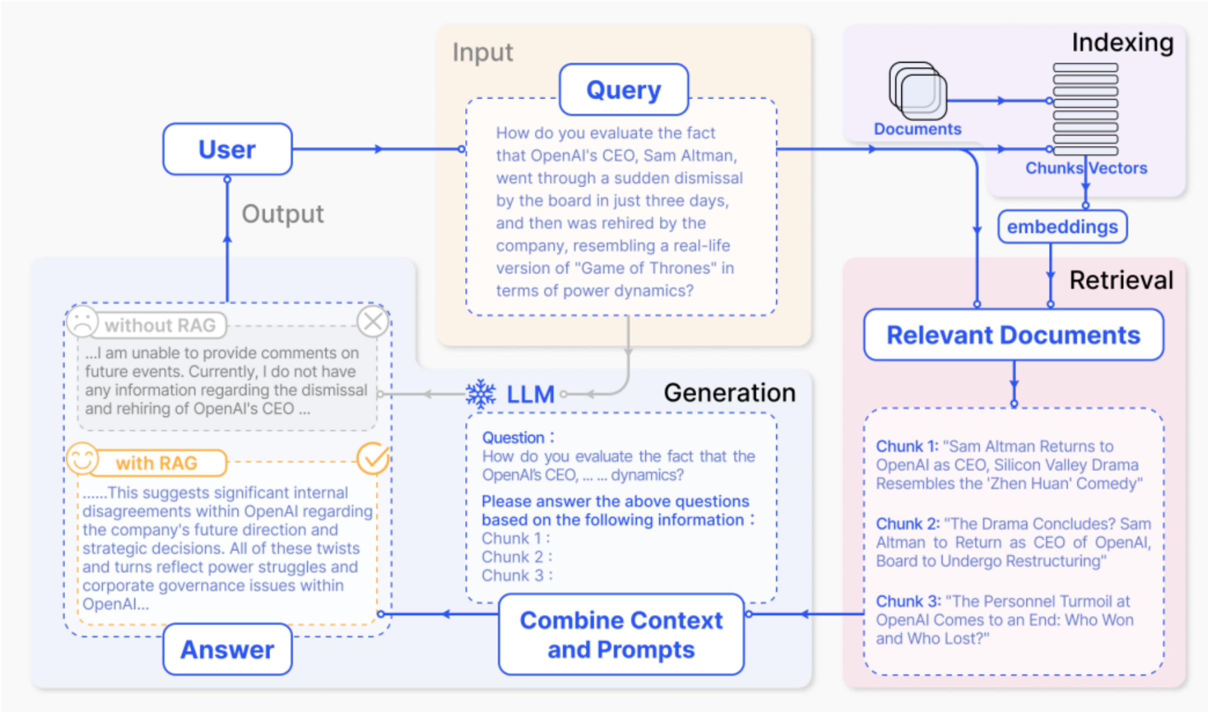


Figure 2.5: A typical representation of a RAG architecture [39]

information, by augmenting the relevant facts along with the initial prompt containing a question as context, resulting in a full prompt that is, then, fed into any generative model, in our context being an LLM. Thus, this architecture allows the model to tackle long-context understanding tasks more effectively by incorporating information retrieval techniques.

By leveraging a retriever, RAG can focus on generating responses based on relevant information rather than relying solely on the pre-trained knowledge stored within the model itself. Thus, RAG effectively reduces the problem of generating factually incorrect content. Its integration into LLMs has resulted in widespread adoption, establishing RAG as a key technology in advancing chatbots and enhancing the suitability of LLMs for real-world applications. Using RAG with a pre-trained model has also proved itself to be a better alternative to fine-tuning the LLM in knowledge-intensive tasks, including question answering [41].

RAGs find themselves in various forms, each of which is better than the others for specific applications and different metrics are used to measure their performance. Out of all the properties that RAGs must have, two of the most important ones are: answer correctness and faithfulness. In the former, the resulting output must align with the ground truth, while the latter concerns itself with whether the answers stick to the facts provided in the context.

2.4.1 Hallucinations

In the context of LLMs, "hallucinations" can be defined as outputs that do not stick to the real-world facts or knowledge encoded in the model parameters. As already mentioned in Section 1, although RAGs help alleviate this problem, they are still prone to producing hallucinations. In the context of RAGs, hallucinations are defined as model outputs that are unfaithful to the retrieved context. A main cause of this is the LLM being more faithful to its training data than to the retrieved context and is more likely to happen when the model has been trained on a much greater volume of text that includes claims inferable from the document or text contradicting the retrieved document. In this project, we aim to directly prevent this from happening by designing prompts that impose the LLM to stick to the provided document.

2.5 Longest common subsequence

For two sequences $\boldsymbol{y} = (y_1, y_2, \dots, y_m)$ and $\boldsymbol{z} = (z_1, z_2, \dots, z_k)$ of tokens/characters, we define a matrix $\boldsymbol{R}$ with elements given by the following recursion:

$$\boldsymbol{R}_{i,j} = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ \boldsymbol{R}_{i-1,j-1} + 1 & \text{if } y_i = z_j \\ \max(\boldsymbol{R}_{i,j-1}, \boldsymbol{R}_{i-1,j}) & o/w. \end{cases} \tag{2.4}$$

Then, the length of the Longest Common Subsequence of $\boldsymbol{y}$ and $\boldsymbol{z}$ is LCS$(\boldsymbol{y}, \boldsymbol{z}) = \boldsymbol{R}_{m,k}$.

For convenience, we also define the following function returning the actual LCS:

$$\text{LCS}^T(\boldsymbol{y}, \boldsymbol{z}) = \begin{cases} \emptyset & \text{if } i = 0 \text{ or } j = 0 \\ \text{LCS}^T\left(\boldsymbol{y}_{[:\|\boldsymbol{y}\|-1]}, \boldsymbol{z}_{[:\|\boldsymbol{z}\|-1]}\right) \oplus y_i & \text{if } y_i = z_j \\ \max\{\text{LCS}^T\left(\boldsymbol{y}, \boldsymbol{z}_{[:\|\boldsymbol{z}\|-1]}\right), \text{LCS}^T\left(\boldsymbol{y}_{[:\|\boldsymbol{y}\|-1]}, \boldsymbol{z}\right)\} & o/w. \end{cases} \tag{2.5}$$

where $\max\{\boldsymbol{y}, \boldsymbol{z}\}$ returns the sequence of larger size.

2.6 Longest common substring

For two sequences $\boldsymbol{y} = (y_1, y_2, \ldots, y_m)$ and $\boldsymbol{z} = (z_1, z_2, \ldots, z_k)$ of tokens/characters, we define a matrix $\boldsymbol{R}$ with elements given by the following recursion:

$$\boldsymbol{R}_{i,j} = \begin{cases} \boldsymbol{R}_{i-1,j-1} + 1 & \text{if } y_i = z_j \\ 0 & o/w. \end{cases} \tag{2.6}$$

Then, the length of the Longest Common Substring of $\boldsymbol{y}$ and $\boldsymbol{z}$ is given by:

$$\text{LCS}\textsc{tr}(\boldsymbol{y}, \boldsymbol{z}) = \max_{\substack{1 \le i \le m \\ 1 \le j \le k}} \boldsymbol{R}_{i,j}$$

Section 3

Design and Implementation

In this section, we formally introduce the problem that is the focus of the project and the notation that we will be using for the remainder of the paper. The LLMs we assume to be working with are decoder-only and fine-tuned for QA.

3.1 Notation

For an LM $\mathcal{M}$, we define, as in Section 2.2.1, $f_{\mathcal{M}} \colon \mathbb{V}^* \to \mathbb{V}^*$ to be a function that maps an input prompt $\boldsymbol{s}$ to a complete response generated by $\mathcal{M}$. Analogously, we define a function $\hat{f}_{\mathcal{M}} \colon [\mathbb{R}^{\mathsf{e}}]^* \to \mathbb{V}^*$ that maps an embedded input prompt $\hat{\boldsymbol{s}}$ to a complete response generated by $\mathcal{M}$, where e is the embedding dimension. In short, we have that $f_{\mathcal{M}}(\boldsymbol{s}) = \hat{f}_{\mathcal{M}}(\hat{\boldsymbol{s}})$.

For brevity, we shall denote $f_{\mathcal{M}}$ and $\hat{f}_{\mathcal{M}}$ by simply f and $\hat{f}$, respectively, as we are not referencing more than one LLM in our work at any point in time. Additionally, whenever we say that an output $\boldsymbol{y}$ has been generated by $\mathcal{M}$ for some input $\boldsymbol{s}$, it can be generated by either applying the input to f or the embedded input $\hat{\boldsymbol{s}}$ to $\hat{f}$ and it should be inferred from the context.

To complete the notation in Section 2.2.1, we also associate to $\mathcal{M}$ a function $\hat{\mathcal{M}} \colon [\mathbb{R}^{\mathsf{e}}]^* \to \mathbb{V}$ such that $\hat{\mathcal{M}}(\hat{\boldsymbol{s}}) = \mathcal{M}(\boldsymbol{s})$.

In the context of question answering, typical RAG systems input relevant content, which we shall denote by **document** $\boldsymbol{d}$, into $\mathcal{M}$ as context alongside a user **query** $\boldsymbol{q}$. This is then used to produce a response or output $\boldsymbol{y}$. We may also include a **prompt** $\boldsymbol{p}$

in the input, thus constructing the input or **full prompt** $\boldsymbol{s}$ that is to be fed into $\mathcal{M}$, which that has the following decomposition:

$$\boldsymbol{s} \triangleq \boldsymbol{d} \oplus \boldsymbol{q} \oplus \boldsymbol{p},$$

for some other permutation of $\{\boldsymbol{d}, \boldsymbol{q}, \boldsymbol{p}\}$, and $\oplus$ denotes concatenation.

In our work, we assume access to a dataset of tuples of the form $(\boldsymbol{q}_z, \boldsymbol{d}_z)$ where z is an index and define $\mathcal{Z}$ to be a uniform distribution over the indexes of the dataset.

The first objective of the project is to find a way to force the output generated by the LLM to be *faithful* to the document $\boldsymbol{d}$ - that is, to contain as much information extractable from $\boldsymbol{d}$ as possible. To this end, we formulate the problem into finding a prompt $\boldsymbol{p}^*$ that is added to the full prompt $\boldsymbol{s}$, such that:

$$\boldsymbol{p}^* = \underset{\boldsymbol{p} \in \mathbb{V}^*}{\operatorname{argmax}} \, \mathbb{E}_{z \sim \mathcal{Z}}[\mathcal{L}(\boldsymbol{d}_z, f(\boldsymbol{s}_z))], \tag{3.1}$$

where $\boldsymbol{s}_z \triangleq \boldsymbol{d}_z \oplus \boldsymbol{q}_z \oplus \boldsymbol{p}^*$ and $\mathcal{L}(\boldsymbol{x}, \boldsymbol{y})$ is some abstract function measuring how much common information do $\boldsymbol{x}$ and $\boldsymbol{y}$ share. In our case, we can take this function to be LCS$(\boldsymbol{x}, \boldsymbol{y})$, which measures the longest common subsequence between $\boldsymbol{x}$ and $\boldsymbol{y}$, as defined in 2.5.

The second objective that we set is to also make sure the output the LLM is giving correctly answers the question. This task is tackled in Section 3.4, where we try to find a prompt $\boldsymbol{p}^{**}$ such that:

$$\boldsymbol{p}^{**} = \underset{\boldsymbol{p} \in \mathbb{V}^*}{\operatorname{argmax}} \, \mathbb{E}_{z \sim \mathcal{Z}}[\lambda \mathcal{L}(\boldsymbol{d}_z, f(\boldsymbol{s}_z)) + (1 - \lambda) \ell(f(\boldsymbol{s}_z), \boldsymbol{q}_z)], \tag{3.2}$$

where $\ell(\boldsymbol{y}, \boldsymbol{q})$ is some abstract function which measures how well a response $\boldsymbol{y}$ answers question $\boldsymbol{q}$ and $\lambda \in [0, 1]$.

3.2 Methodology

As we will see in Section 4, manually searching for a $\boldsymbol{p}^*$ that satisfies condition 3.1 doesn't significantly improve the values of the metrics for measuring how *faithful* the answer is to the document. To this end, we develop a framework that leverages, in Section 3.3, the memory efficiency and good performance of Prompt tuning [32] in order to optimize over the continuous space of token representations and find a $\hat{\boldsymbol{p}}^*$ satisfying 3.1, by minimizing a loss function which measures how *faithful* an answer is to the document.

We then turn our attention to the second objective of the project in Section 3.4, where we also want the answer to correctly answer the question. We propose using the cross-entropy objective and balancing it out with the loss function used to train the prompt embeddings $\hat{\boldsymbol{p}}$ into forcing the model to output as much information from the given document as possible. We describe the algorithm for training the soft prompts in the final architecture in Section 3.5.

To evaluate the performance of the optimized prompts, we implement some discrete metrics which we test the prompts $\hat{\boldsymbol{p}}$ against. We analyze their implementation in Section 3.6 and talk about the experiments in Section 4.

3.2.1 Technologies used

We provide scripts written in *Python 3.10* for all the models we designed and the experiments we've produced. To achieve these results, several libraries were used. All tensor and gradient computations were done using *PyTorch* [42]. For loading and batching datasets, the *HuggingFace's datasets* [43] library was used, along with the *transformers* [44] library for loading transformer-based language models, performing inferences, tokenizing inputs and computing the logits for each of its tokens. For optimizing over soft prompts using Prompt tuning, we make use of the *peft* [45] library, which contains multiple PEFT methods implemented for a various number of base LMs.

3.3 "Differentiable" LCS

Let $\boldsymbol{y} = (y_1, y_2, \dots, y_k)$ and $\boldsymbol{d} = (d_1, d_2, \dots, d_m)$ be two sequences of tokens with their longest common subsequence LCS$(\boldsymbol{d}, \boldsymbol{y})$ as defined in 2.5. Assume $\boldsymbol{y}$ to be the model $\mathcal{M}$'s output for some input $\boldsymbol{s}$ and $\boldsymbol{d}$ the document returned by a RAG system. Although this function would work well as a loss function for our purposes, it is clearly not differentiable with respect to model parameters:

$$\text{LCS}(\boldsymbol{d}, \boldsymbol{y}) = \boldsymbol{R}_{m,k} = \chi(d_m, y_k)\left(\boldsymbol{R}_{m-1,k-1} + 1\right) + \left(1 - \chi(d_m, y_k)\right)\max\left(\boldsymbol{R}_{m,k-1}, \boldsymbol{R}_{m-1,k}\right),$$

where $\chi(d_i, y_j) = \begin{cases} 1, & \text{if } d_i = y_j, \\ 0, & \text{otherwise.} \end{cases} \quad \forall 1 \leq i \leq k, 1 \leq j \leq k$

Then, we have that:

$$\frac{\partial \text{LCS}(\boldsymbol{d}, \boldsymbol{y})}{\partial \Phi} = \frac{\partial \chi(d_m, y_k)}{\partial \Phi}\left(\boldsymbol{R}_{m-1,k-1} + 1\right) + \chi(d_m, y_k)\frac{\partial\left(\boldsymbol{R}_{m-1,k-1} + 1\right)}{\partial \Phi} +$$

$$+ \frac{\partial\left(1 - \chi(d_m, y_k)\right)}{\partial \Phi}\max\left(\boldsymbol{R}_{m,k-1}, \boldsymbol{R}_{m-1,k}\right) + \left(1 - \chi(d_m, y_k)\right)\frac{\partial \max\left(\boldsymbol{R}_{m,k-1}, \boldsymbol{R}_{m-1,k}\right)}{\partial \Phi}$$

Clearly, $\chi(d_i, y_j)$ is a token comparison function that is not continuous on all points in $\mathbb{R}^{\|\Phi\|}$ where Φ denotes $\mathcal{M}$'s parameters that determine the output $\boldsymbol{y}$.

CaLCS [46] is a function that was originally designed to be used as a loss function for Seq2Seq models in order to improve their suboptimal performance induced by the fact that they are trained in an autoregressive manner, where we update the model's parameters at each 'next-token' prediction, while at test time, the model generates a whole sequence, where each newly predicted token is predicted based on the previously generated tokens. This discrepancy is also referred to as *exposure bias* [47], which is one of the factors leading LLMs to hallucinate. The paper, thus, proposes to leverage a sequence-level loss objective function that closely resembles the (non-differentiable) discrete metrics the models are evaluated on, such as *BLEU* [48], for text translation, or *ROUGE* [49], for summarization.

CALCS (Continuous Approximation of Longest Common Subsequence) is a continuous approximation to the longest common subsequences between two sequences of tokens that can be directly optimized against using standard gradient-based methods. Informally, it relaxes the hard inferences while computing discrete metrics and thus compares them in a soft way.

In addition to the variables defined above, let $p_1, p_2, \ldots, p_k$ be the probability distributions over vocabulary $\mathbb{V}$ at the decoding time steps from which $y_1, y_2, \ldots, y_k$ are generated by $\mathcal{M}$. The idea is to recursively define the *soft LCS* $\boldsymbol{S}_{i,j}$ between prefixes $\boldsymbol{d}_{[:i]}$ and $\boldsymbol{y}_{[:j]}$. Then, consider the relaxed version of the *LCS*, namely CALCS, between $\boldsymbol{y}$ and $\boldsymbol{d}$:

$$\boldsymbol{S}_{i,j} = p_j^{(d_i)}(\boldsymbol{S}_{i-1,j-1} + 1) + (1 - p_j^{(d_i)}) \max(\boldsymbol{S}_{i,j-1}, \boldsymbol{S}_{i-1,j}) \tag{3.3}$$

for $i, j > 0$ and $\boldsymbol{S}_{i,0} = \boldsymbol{S}_{0,j} = 0$, where $p_j^{(d_i)}$ denotes the probability of generating d_i at $j-th$ decoding step. In essence, the hard token comparison function $\chi(y_i, d_j)$ was replaced by the probability $p_j^{(d_i)}$.

Then CALCS$(\boldsymbol{d}, \boldsymbol{y}) = \boldsymbol{S}_{m,k}$.

Now, let $l_1, l_2, \ldots, l_k$ be the unnormalized logits of the model output before applying SOFTMAX to obtain probabilities $p_1, p_2, \ldots, p_k$ at decoding time steps, respectively. Then:

$$p_j^{(t_i)} = \frac{\exp(l_j^{(t_i)})}{\sum_{o=1}^{\|\mathbb{V}\|} \exp(l_j^{(t_o)})} \quad \forall j = 1, 2, \ldots, k$$

Here, t_i is the token with index i in the vocabulary $\mathbb{V}$. If we also add a temperature hyperparameter α, we obtain *peaked* SOFTMAX:

$$p_j^{(t_i)}(\alpha) = \frac{\exp(l_j^{(t_i)}/\alpha)}{\sum_{o=1}^{\|\mathbb{V}\|} \exp(l_j^{(t_o)}/\alpha)} \quad \forall j = 1, 2, \ldots, k \tag{3.4}$$

If we additionally define $\delta_{i,j} = \boldsymbol{S}_{i,j} - \boldsymbol{R}_{i,j}$ to be the approximation error, then, according to [46], we can control the error for a CALCS loss function when using peaked SOFTMAX:

$$|\delta_{i,j}| \to 0 \quad \text{as} \quad \alpha \to 0$$

Consider an embedded batch of size of size N $\{(\boldsymbol{q}_i, \boldsymbol{d}_i)\}_{0<i<N}$ and the sequence of outputs $\{\boldsymbol{y}_i\}$ generated by frozen LM $\mathcal{M}$ on the full inputs $\hat{\boldsymbol{s}}_i \triangleq \hat{\boldsymbol{p}} \oplus \hat{\boldsymbol{d}}_i \oplus \hat{\boldsymbol{q}}_i$, for some trainable prompt embeddings $\hat{\boldsymbol{p}}$. The paper [46] proposes the following loss function, which has been adapted for our use case and can be used as a good candidate from the abstract loss function in 3.1:

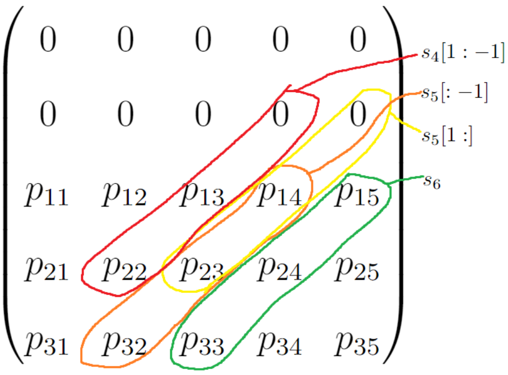
$$\mathcal{L}_{\text{CALCS}}(\hat{\boldsymbol{p}}) = -\frac{1}{N}\sum_{i=1}^{N} \log\left(\frac{\text{CALCS}(\boldsymbol{d}_i, \boldsymbol{y}_i)}{|\boldsymbol{d}_i|}\right), \tag{3.5}$$

where $|\boldsymbol{d}_i|$ is the size of the tokenized document $\boldsymbol{d}_i$

Implementation

The trivial algorithm which performs a $\mathcal{O}(m \cdot k)$ sequential steps in order to compute $\boldsymbol{S}_{m,k}$ proved to be impractical, as just for one sample with a reasonably-sized document of 10^3 tokens and answer of 200 tokens it would take about 1 minute to compute. To mitigate this issue, a vectorized algorithm with just $\mathcal{O}(m + k)$ sequential steps, where we can leverage the power of GPU parallelization to compute the loss in feasible time, was designed and implemented.

The core idea behind the algorithm is that when computing $\boldsymbol{S}_{i,j}$, which is situated on some anti-diagonal of the matrix $\boldsymbol{S}$, we are only referencing some elements from the



$$s_6 = s_6 \cdot (s_4[1:-1] + \mathbf{1}) + (\mathbf{1} - s_6) \cdot \max(s_5[:-1], s_5[1:])$$

Figure 3.1: An example of how anti-diagonal s_6 in the CALCS algorithm is computed. We assume s is initialized with the values from p. All operations are computed element-wise.

previous two anti-diagonals - namely, the former term from the summation in Eq. 3.3 is situated on the anti-diagonal that is before the previous, while the latter contains two matrix entries situated on the previous anti-diagonal. To exploit this property, we iteratively compute the anti-diagonals of $\boldsymbol{S}$ until the last, which contains only the entry $\boldsymbol{S}_{m,k}$. For the sake of explanation, we first pad the matrix $p = [p_1, p_2, \ldots, p_k]$ containing the probability distributions with 0 values to the left or upwards so that it becomes a square matrix (i.e., we now have $m = k$). For example, suppose $p \in \mathbb{R}^{3\times5}$. Then, padding is done as follows:

$$\begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} & p_{15} \\ p_{21} & p_{22} & p_{23} & p_{24} & p_{25} \\ p_{31} & p_{32} & p_{33} & p_{34} & p_{35} \end{pmatrix} \rightarrow \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ p_{11} & p_{12} & p_{13} & p_{14} & p_{15} \\ p_{21} & p_{22} & p_{23} & p_{24} & p_{25} \\ p_{31} & p_{32} & p_{33} & p_{34} & p_{35} \end{pmatrix}$$

The algorithm, which is also depicted in Figure 3.1, has a pseudocode in Algorithm 1.

The pseudocode of Algorithm 1 is written in a *Python* style to focus on the tensor-wise operations that take place and which can be computed in a multi-core environment

```
Algorithm 1: CALCS Parallel Algorithm
Input : p
p.flip(axis = 0) // flips the matrix w.r.t axis 0 so that now anti-diagonals become diagonals
x_0 ← [p[0, 0]]
x_1 ← []
for i ← 1 to 2·m − 1 do
    d ←p.diag(i) // gets the i − th diagonal of p as a vector
    ones ←[1] ·d.size(0) // constructs a vector of the same size as d filled with 1s
    // x_0 and x_1 resemble the (i − 1) − th and (i − 2) − th diagonals of S, respectively
    if i < m then
        x_1 ←[0, 0] ⊕x_1 ⊕ [0, 0]
        x_0 ←[0] ⊕x_0 ⊕ [0]
    // max(·)computeselement − wisemaximumbetweentheelementsofthe2vectors
    mx ← max(x_0[: −1], x_0[1 :])
    sc ← (ones − d) · mx  // subtraction and multiplication are computed element-wise between the elements of the 2 vectors
    ones1 ←[1] ·x_1.size(0)  // constructs a vector of the same size as x_1 filled with 1s
    x_1 ← (x_1 + ones1)  // addition is computed element-wise
    if i > m then
        x_1 ← x_1[1 : −1]
    pc ←d ·x_1  // multiplication is computed element-wise
    x_1 ← x_0
    x_0 ← pc + sc
    // addition is computed element-wise between the elements of the 2 vectors
end return x_0
```

using libraries like *torch*. In particular, all functions used in the pseudocode have an implementation in *torch*. The official implementation, which can be found in the project's *github* repository, is adapted to using batches and is coded in a manner where we don't perform the process of padding the matrix p. Thus, the official implementation contains a lot of edge cases to be taken into consideration and couldn't benefit from a brief explanation as the algorithm we showed above, yet the essence remains the same and both implementations are equivalent.

3.4 Joint loss

As we have already argued, the CALCS loss function is a very promising instance of a loss function that we can use to tackle the first task of this project, namely 3.1. It was defined in a setting where our LM's model parameters are frozen and we optimize over some prompt embeddings $\hat{\boldsymbol{p}}$ via Prompt tuning.

As we will see in Section 4.3, this approach gives positive results and manages to find such $\hat{\boldsymbol{p}}$ which, if we append to any input, the model $\mathcal{M}$, whose parameters have not been updated at all, manages to give an answer to the question in the input that contains as much verbatim information from the document in the input as possible. Unsurprisingly, though, the answer will not be relevant to the question at all most of the time, as so far we have been mainly focusing on getting an answer that is textually similar to the document.

To tackle this issue, we need to find a good loss function that is a good candidate for the ℓ abstract function in 3.2. A good approach is to simply use the cross-entropy loss function used when pre-training an autoregressive model, as in Def.2.3, adapted for fine-tuning for the QA task.

Formally, consider a batch of size N, $\boldsymbol{B} = \{(\boldsymbol{q}_i, \boldsymbol{d}_i)\}_{1 \leq i \leq N}$, and the sequence of outputs $\{\boldsymbol{y}_i\}$ generated by a frozen LM $\mathcal{M}$ for some prompt embeddings $\hat{\boldsymbol{p}}$. We assume that the samples in the batch $\boldsymbol{B}$ come from a dataset $\mathcal{D}$ for question answering. Thus, let $\boldsymbol{a}_i$ be the ground-truth answer for input $\hat{\boldsymbol{s}}_i \triangleq \hat{\boldsymbol{d}}_i \oplus \hat{\boldsymbol{q}}_i \oplus \hat{\boldsymbol{p}}$. Let $\mathcal{P}\left(\cdot | \hat{\boldsymbol{s}}_i \oplus \hat{\boldsymbol{a}}_{i[:j-1]}\right)$ be the probability distribution outputted by the LLM for the j-th token of $\boldsymbol{a}_i$ $\forall j \in \{1, \ldots, \|\boldsymbol{a}_i\|\}$, we define the loss function as:

$$\mathcal{L}_{CE}(\hat{\boldsymbol{p}}) = \frac{1}{N} \sum_{i=1}^{N} \left(-\frac{1}{\|\boldsymbol{a}_i\|} \sum_{j=1}^{\|\boldsymbol{a}_i\|} \log \left(\mathcal{P}\left(\boldsymbol{a}_{i[j]} | \hat{\boldsymbol{s}}_i \oplus \hat{\boldsymbol{a}}_{i[:j-1]} \right) \right) \right) \tag{3.6}$$

We shall now define the *Joint loss* objective which we shall use for approximately optimizing for a $\hat{\boldsymbol{p}}$ such that the condition in 3.2 is met:

$$\mathcal{L}(\hat{\boldsymbol{p}}) = \lambda \mathcal{L}_{\text{CaLCS}}(\hat{\boldsymbol{p}}) + (1 - \lambda) \mathcal{L}_{CE}(\hat{\boldsymbol{p}}), \tag{3.7}$$

where λ is a hyperparameter.

This function aims at balancing the objective of getting an answer that is as similar to the document given in the input as possible and the one of getting a correct answer to the question in the input.

3.5 Training

Now that we defined a reasonable loss function that we can use to achieve our objectives, we shall describe in detail the training algorithm of one iteration given a batch $\boldsymbol{B}$ of inputs. We assume the same setup as described in the above section:

i. Compute $\boldsymbol{y}_i = \hat{f}(\hat{\boldsymbol{s}}_i)$ of length L_i and its logits $\quad \forall 1 \leq i \leq N$

ii. Compute $\mathcal{L}_{\textsf{CALCS}}(\hat{\boldsymbol{p}})$

iii. Compute the probability distributions $\mathcal{P}\left(\boldsymbol{a}_{i[j]} | \hat{\boldsymbol{s}}_i \oplus \hat{\boldsymbol{a}}_{i[:j-1]}\right) \quad \forall 1 \leq i \leq N, \quad j \in \{1, \ldots, K_i\}$, where $K_i = \|\boldsymbol{a}_i\|$

iv. Compute $\mathcal{L}_{CE}(\hat{\boldsymbol{p}})$ and $\mathcal{L}(\hat{\boldsymbol{p}})$

v. Compute $\frac{\partial \mathcal{L}_{\textsf{CaLCS}}(\hat{\boldsymbol{p}})}{\partial \hat{\boldsymbol{p}}} = -\frac{1}{N} \sum_{i=1}^{N} \frac{\partial \log\left(\frac{\textsf{CALCS}(\boldsymbol{d}_i, \boldsymbol{y}_i)}{|\boldsymbol{d}_i|}\right)}{\partial \boldsymbol{y}_i} \frac{\partial \boldsymbol{y}_i}{\partial \hat{\boldsymbol{p}}} =$

$$= -\frac{1}{N} \sum_{i=1}^{N} \frac{\partial \log\left(\frac{\textsf{CALCS}(\boldsymbol{d}_i, \boldsymbol{y}_i)}{|\boldsymbol{d}_i|}\right)}{\partial \boldsymbol{y}_i} \frac{\partial \hat{\mathcal{M}}\left(\hat{\boldsymbol{s}} \oplus \hat{\boldsymbol{y}}_{i[:L_i-1]}\right)}{\partial \boldsymbol{y}_{i[:L_i-1]}} \frac{\partial \boldsymbol{y}_{i[:L_i-1]}}{\partial \hat{\boldsymbol{p}}} =$$

$$= -\frac{1}{N} \sum_{i=1}^{N} \frac{\partial \log\left(\frac{\textsf{CALCS}(\boldsymbol{d}_i, \boldsymbol{y}_i)}{|\boldsymbol{d}_i|}\right)}{\partial \boldsymbol{y}_i} \frac{\partial \hat{\mathcal{M}}\left(\hat{\boldsymbol{s}} \oplus \hat{\boldsymbol{y}}_{i[:L_i-1]}\right)}{\partial \boldsymbol{y}_{i[:L_i-1]}} \frac{\partial \hat{\mathcal{M}}\left(\hat{\boldsymbol{s}} \oplus \hat{\boldsymbol{y}}_{i[:L_i-2]}\right)}{\partial \boldsymbol{y}_{i[:L_i-2]}} \frac{\partial \boldsymbol{y}_{i[:L_i-2]}}{\partial \hat{\boldsymbol{p}}} =$$

$$= \cdots =$$

$$= -\frac{1}{N} \sum_{i=1}^{N} \frac{\partial \log\left(\frac{\textsf{CALCS}(\boldsymbol{d}_i, \boldsymbol{y}_i)}{|\boldsymbol{d}_i|}\right)}{\partial \boldsymbol{y}_i} \frac{\partial \hat{\mathcal{M}}\left(\hat{\boldsymbol{s}} \oplus \hat{\boldsymbol{y}}_{i[:L_i-1]}\right)}{\partial \boldsymbol{y}_{i[:L_i-1]}} \frac{\partial \hat{\mathcal{M}}\left(\hat{\boldsymbol{s}} \oplus \hat{\boldsymbol{y}}_{i[:L_i-2]}\right)}{\partial \boldsymbol{y}_{i[:L_i-2]}} \frac{\partial \hat{\mathcal{M}}\left(\hat{\boldsymbol{s}} \oplus \hat{\boldsymbol{y}}_{i[:L_i-3]}\right)}{\partial \boldsymbol{y}_{i[:L_i-3]}} \cdots \frac{\boldsymbol{y}_{i[:1]}}{\hat{\boldsymbol{p}}}$$

vi. Compute $\frac{\partial \mathcal{L}_{CE}(\hat{\boldsymbol{p}})}{\partial \hat{\boldsymbol{p}}} = -\frac{1}{N} \sum_{i=1}^{N} \left(\frac{1}{\|\boldsymbol{a}_i\|} \sum_{j=1}^{\|\boldsymbol{a}_i\|} \frac{\partial \log\left(\mathcal{P}\left(\boldsymbol{a}_{i[j]} | \hat{\boldsymbol{s}}_i \oplus \hat{\boldsymbol{a}}_{i[:j-1]}\right)\right)}{\partial \hat{\boldsymbol{p}}} \right)$

vii. Compute the joint loss $\frac{\partial \mathcal{L}(\hat{\boldsymbol{p}})}{\partial \hat{\boldsymbol{p}}} = \lambda \frac{\partial \mathcal{L}_{\textsf{CaLCS}}(\hat{\boldsymbol{p}})}{\partial \hat{\boldsymbol{p}}} + (1-\lambda) \frac{\partial \mathcal{L}_{CE}(\hat{\boldsymbol{p}})}{\partial \hat{\boldsymbol{p}}}$

viii. Perform backpropagation and update $\hat{\boldsymbol{p}}$

Implementation

The gradient computations in step v and step vi are handled by PyTorch's *autograd*, which is an automatic differentiation engine. During the forward pass and loss computations, records of all the operations involving model parameters are stored in a Directed Acyclic Graph, called the Computational Graph, with the root being the value of the joint loss $\mathcal{L}(\hat{\boldsymbol{p}})$. During backpropagation, the resulting Computational Graph is traversed starting from the root, with gradients being accumulated for each parameter. After backpropagation, the parameters are updated using the *Adam* [50] optimization algorithm based on gradient descent.

In order to compute the cross-entropy loss function at step iv and logits at step i, we perform the inference on the given inputs by calling the *forward* method of the model instance loaded using the *peft* library. The returned value contains the logits for each token and the cross-entropy loss value based on only the tokens of the answer, which can be done by masking the input $\hat{\boldsymbol{s}}_i \oplus \hat{\boldsymbol{a}}_i$ that is passed onto the forward method.

Generating the output in step i is done by generating the tokens in an autoregressive manner using the *forward* method of the model, as shown in Section 2.2.1:

$$\boldsymbol{y} = \hat{f}(\hat{\boldsymbol{s}}) \triangleq y_{d_{out}} \oplus \cdots \oplus y_1, \text{where } y_i = \hat{\mathcal{M}}\left(\hat{\boldsymbol{s}} \oplus \hat{\boldsymbol{y}}_{[:i-1]}\right) \quad \forall i \in \{1, \ldots, d_{out}\}$$

Batched generation of the outputs in step i is done by padding the tokenized inputs to the left with a special padding token and the attention masks with 0s, so that the padding tokens get ignored during the attention matrices computation.

KV-caching

Because the model is autoregressive, when computing the masked unnormalized attention matrix, each token y_i will only attend to the tokens before it, namely $\{y_j\}_{1 \leq j < i}$. Thus, when computing the tokens $\{y_j\}_{y_{d_{out}} \geq j > i}$, the attention and, therefore, the hidden state values of y_i, are recomputed, when they actually don't change at all throughout the whole generation. This creates a massive overhead, something which can by mitigated by caching the key $\boldsymbol{K}$ and value $\boldsymbol{V}$ matrices passed onto each Multi-Head Attention layer of the previously generated tokens at the cost of additional memory usage, a technique that is called *Key-Value (KV) caching*. The base models that we used in our implementation have this technique implemented in their Masked Multi-Head Attention Mechanism as such:

Suppose token y_i was just generated and has been concatenated to the current sequence of generated tokens $\boldsymbol{y}_{[:i]}$. We save the key and value matrices $\boldsymbol{K}_{l,j}$ and $\boldsymbol{V}_{l,j}$, respectively, at layer (i.e. decoder block) l of each token y_j in $\boldsymbol{y}_{[:i-1]}$ that has been computed when generating y_i into matrices $\boldsymbol{K}_{l,j}^{prev}$ and $\boldsymbol{V}_{l,j}^{prev}$. Now, when computing $y_{i+1} = \hat{\mathcal{M}}\left(\hat{\boldsymbol{s}} \oplus \hat{\boldsymbol{y}}_{[:i]}\right)$, at Multi-Head Attention Mechanism found at layer l with queries, keys and values $\boldsymbol{Q}_l$ and $\boldsymbol{V}_l$, we set $\boldsymbol{Q} = \boldsymbol{Q}_{l,i}$, $\boldsymbol{K} = \boldsymbol{K}_l^{prev} \oplus_0 \boldsymbol{K}_{l,i}$ and $\boldsymbol{V} = \boldsymbol{V}_l^{prev} \oplus_0 \boldsymbol{V}_{l,i}$ in Def. 2.2, where $\oplus_0$ is concatenation row-wise. This way, only the attentions for the newly added token are computed and the running-time of the attention mechanism becomes quadratic instead of cubic.

3.6 Metrics

RAG systems are evaluated based on the quality of the retriever and generator. There are several available benchmarks for evaluating RAG systems, such as *RAGAs* [51] and *ARES* [52], each of them focusing on assessing different aspects of these components. Since our work is meant to enhance the generator component of a RAG, we are interested in the two main metrics typically used to evaluate this component, namely *Answer Relevancy* and *Faithfulness*.

Faithfulness measures how *faithful* an answer of an LLM is to the document provided as context. In other words, it quantifies whether the output factually aligns with the claims in the document. In most benchmarks, this score is computed using LLMs, thus leveraging their understanding of language to provide high-quality assessments. Such an example is using an *"LLM as a Judge"* [53], a method where an LLM is used to predict a score for an answer based on how *faithful* it is to a given document.

Since, in our setting, we aim for the LLM outputting verbatim statements from the document rather than, simply, answers that are factually aligned to the document, we propose two discrete metrics for measuring how well a model answer to a question extracts information from an augmented document rather than use the standard faithfulness metrics. Firstly, this design decision is based on the hypothesis that guiding an LLM for a more restrictive task such as outputting verbatim text may show better results in terms of our objective. Furthermore, these metrics resemble the CALCS functionality better than the standard metrics do.

In our setting, we aim for the LLM outputting verbatim statements from the document rather than, simply, answers that are factually aligned to the document. Hence we propose two discrete metrics for measuring how well a model answering a question extracts information from an augmented document rather than using the standard faithfulness metrics. We make this design choice as it not only suits the objective, but these metrics also resemble our proposed CALCS function better than the standard metrics do.

The third metric we use is, thus, answer relevancy, which is a standard metric for measuring whether the generated answers actually address the queries in the input or not.

3.6.1 Exact String Matching

Let $\text{CSTR}_\tau(\boldsymbol{x}, \boldsymbol{y})$ be the set of common subtrings between strings $\boldsymbol{x}$ and $\boldsymbol{y}$ that contain at least τ characters such that for each $\boldsymbol{s} \in \text{CSTR}_\tau(\boldsymbol{x}, \boldsymbol{y})$, there is no other common substring $\boldsymbol{s}'$ between $\boldsymbol{x}$ and $\boldsymbol{y}$ such that $\boldsymbol{s} \subseteq \boldsymbol{s}'$. Then, we define the exact string matching score between two strings $\boldsymbol{x}$ and $\boldsymbol{y}$ as such:

$$\text{EXACT}_\tau(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{\|\boldsymbol{x}\|^2} \sum_{\boldsymbol{w} \in \text{CSTR}_\tau(\boldsymbol{x}, \boldsymbol{y})} \|\boldsymbol{w}\|^2 \tag{3.8}$$

Given a model output $\boldsymbol{y}$ that answers a question with augmented document $\boldsymbol{d}$, the metric $\text{EXACT}_\tau(\boldsymbol{y}, \boldsymbol{d})$, which can only take values in the range $[0, 1]$, is meant to quantify how much of the output consists of verbatim information extracted from the document.

Implementation

$\text{EXACT}_\tau(\boldsymbol{x}, \boldsymbol{y})$ is computed by repeatedly extracting the longest common substring between strings $\boldsymbol{x}$ and $\boldsymbol{y}$, as defined in Section 2.6. We note that this metric is computed at the level of characters rather than tokens as we have mainly considered in the report so far. Whenever we compute $\boldsymbol{l} = \text{LCSTR}(\boldsymbol{x}, \boldsymbol{y})$, we replace the substring $\boldsymbol{l}$ from $\boldsymbol{x}$ with some separation character (and thus, we remove it) and repeat the process until there is no longer a common substring that is longer than τ, for which we set a value of 12.

In order to compute $\text{LCSTR}(\boldsymbol{x}, \boldsymbol{y})$, we make use of the Python library *pylcs* [54] which is a fast library written in C++ for efficiently computing LCSTR and LCS.

3.6.2 Fuzzy String Matching

Fuzzy string matching is a metric originally proposed in [55] to evaluate how much of an output given by an LLM can be interpreted as a literal reproduction of copyright material. The paper [55] defines the metric, but offers an incomplete design and implementation of the fuzzy string matching algorithm. In this project, we use the definitions proposed in the paper, while we adapt this metric for our use case and implement an algorithm for computing this score.

Definition Given two sequences of tokens $\boldsymbol{x}$ and $\boldsymbol{y}$ and $\boldsymbol{z} = \text{LCS}(\boldsymbol{x}, \boldsymbol{y})$, we say that $\boldsymbol{x}$ and $\boldsymbol{y}$ are *similar* if and only if the following conditions hold:

- $\forall \mathrm{x}_i \in \boldsymbol{x}$: if $x_i \notin \boldsymbol{z}$, then $x_{i-1} \in \boldsymbol{z}$ and $x_{i-2} \in \boldsymbol{z}$
- $\forall \mathrm{y}_i \in \boldsymbol{x}$: if $y_i \notin \boldsymbol{z}$, then $y_{i-1} \in \boldsymbol{z}$ and $y_{i-2} \in \boldsymbol{z}$

Next, we give the definition of a *match* for $\boldsymbol{y}$ and $\boldsymbol{d}$:

Definition Given strings $\boldsymbol{x}$ and $\boldsymbol{y}$ and a length threshold τ, substrings $\boldsymbol{z}$ of $\boldsymbol{x}$ and $\boldsymbol{z}'$ of $\boldsymbol{y}$ form a *match* $\boldsymbol{w} = \text{LCS}^T(\boldsymbol{z}, \boldsymbol{z}')$ for $\boldsymbol{x}$ and $\boldsymbol{y}$ if and only if the following conditions are met:

- $\boldsymbol{z}$ and $\boldsymbol{z}'$ are similar
- $\|\boldsymbol{w}\| > \tau$

The *Fuzzy Threshold Common Substring Problem* is to find all matches for two sequences of tokens $\boldsymbol{x}$ and $\boldsymbol{y}$ such that there is no other match $\boldsymbol{w}'$ for $\boldsymbol{x}$ and $\boldsymbol{y}$ such that $\boldsymbol{w} \subseteq \boldsymbol{w}'$. We define the set of such matches as $\text{MATCH}_\tau(\boldsymbol{x}, \boldsymbol{y})$. We then, finally, define the following fuzzy string matching metric:

$$\text{FUZZY}_\tau(\boldsymbol{x}, \boldsymbol{y}) = \sum_{\boldsymbol{w} \in \text{MATCH}_\tau(\boldsymbol{x}, \boldsymbol{y})} \|\boldsymbol{w}\|^2 \tag{3.9}$$

Informally, for a model $\mathcal{M}$ output $\boldsymbol{y}$ to a question based on a document $\boldsymbol{d}$, $\text{FUZZY}_\tau(\boldsymbol{y}, \boldsymbol{d})$ measures how many subsequences of $\boldsymbol{y}$ are part of a literal reproduction of the document, where the longer the subsequence, the more weight it has in the final score. In other words, this metric helps us figure out if an LM is more willing to answer based on the information found in the augmented documents than on the information from its parameters.

Design

Given two sequences of tokens $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ and $\boldsymbol{y} = (y_1, y_2, \ldots, y_m)$, and construct nodes $\mathcal{N}_{x_i,y_j}$ for each pair of tokens $x_i \in \boldsymbol{x}$ and $y_j \in \boldsymbol{y}$ such that $x_i = y_j$, then iteratively go through each $x_i \in \boldsymbol{x}$ for $\forall 1 \leq i \leq n$. The idea behind the algorithm for *Fuzzy Threshold Common Substring Problem* is to satisfy the following condition for each each node $\mathcal{N}_{x_i,y_j}$ at the end of iteration i:

$(*)$ Node $\mathcal{N}_{x_i,y_j}$ is connected to a tree having previous nodes $\mathcal{N}_{x_k,y_t}$ with $k < i$ so that the path from $\mathcal{N}_{x_i,y_j}$ to the root tree is the largest possible match for $\boldsymbol{x}_{[:i]}$ and $\boldsymbol{y}$ ending at positions i and j.

Suppose we are at iteration i and all nodes $\mathcal{N}_{x_k,y_t}$ with $k < i$ satisfy the condition. Let $\mathcal{N}_{x_i,y_j}$ be the current node for which we are trying to find a parent $\mathcal{N}_p$. There are 4 cases that must be taken into account in the given order of importance:

i. There is a node $\mathcal{N}_{x_{(i-1)},y_{(j-1)}}$. In this case, we pick this to be the parent.

ii. There is a node $\mathcal{N}_{x_{(i-1)},y_{(j-2)}}$ that is connected to an existing parent $\mathcal{N}_{x_k,y_{(j-3)}}$ or $\mathcal{N}_{x_{(i-2)},y_{(j-1)}}$ that is connected to an existing parent $\mathcal{N}_{x_{(i-3)},y_k}$. If both nodes exist, we pick the one that has a larger path between itself and the root to be the parent. Otherwise, we pick the one that exists.

iii. There is a node $\mathcal{N}_{x_{(i-2)},y_{(j-2)}}$ that is connected to an existing node $\mathcal{N}_{x_{(i-3)},y_{(j-3)}}$. In this case, we pick this to be the parent.

iv. Parent is $\emptyset$

If a parent is found, we create an edge $(\mathcal{N}_{x_i,y_j}, \mathcal{N}_p)$. Otherwise, $(\mathcal{N}_{x_i,y_j}, \mathcal{N}_p)$ is the root of a new tree. Then, the condition stated above is satisfied for node $\mathcal{N}_{x_i,y_j}$.

At the end of this process, all nodes $\mathcal{N}_{x_i,y_j}$ will be part of some tree. We go through all nodes $\mathcal{N}_{x_i,y_j}$ that are leaves and record the path from them to the root of the tree they are part of. Then, all such paths make up the set $\textsc{Match}_\tau(\boldsymbol{x}, \boldsymbol{y})$. Finally, we simply compute $\textsc{Fuzzy}_\tau(\boldsymbol{x}, \boldsymbol{y})$.

Implementation

To implement the following algorithm, an approach is to create a class for a node $\mathcal{N}_{x_i,y_j}$ which holds information on the index in $\boldsymbol{x}$ and the index in $\boldsymbol{y}$ and a reference to its parent. We can hold a set of sets $\mathcal{S}$, where $\mathcal{S}[i]$ stores all nodes $\mathcal{N}_{x_i,y_j}$ satisfy the condition. When processing node $\mathcal{N}_{x_i,y_j}$ (i.e. looking for a parent to connect to), we check for the existence of nodes in $\mathcal{S}[i-1]$ and $\mathcal{S}[i-2]$. if each set $\mathcal{S}[i-1]$ is implemented as a vector, the search for a node in such a set is done in $\mathcal{O}(m)$ running time. Another approach is to have it implemented as a binary search tree, in which case the running time for performing a search becomes $\mathcal{O}(log(m))$. We use $\tau = 4$ in our implementation.

3.6.3 Answer relevancy

There are multiple ways benchmarks implement answer relevancy. For instance, given a question $\boldsymbol{q}$ and an answer $\boldsymbol{y}$ generated by an LM $\mathcal{M}$, *RAGAs* leverages other LLMs by asking them to generate possible questions that could have outputted $\boldsymbol{y}$ to $\boldsymbol{q}$. It then collects all the generated possible questions and averages the sum of cosine similarities between the embeddings of the generated questions and the actual question $\boldsymbol{q}$.

In our project, we use the answer relevancy metric, which we shall denote by $\textsc{AnsRel}(\boldsymbol{y}, \boldsymbol{q})$, implemented by *Deepval* [56], an LLM evaluation framework that provides a Python library for computing RAG/LLM evaluation metric using helper LLMs that can be locally stored in memory. In their implementation, this metric is computed as follows:

$$\textsc{AnsRel}(\boldsymbol{y}, \boldsymbol{q}) = \frac{\texttt{Number of Relevant Statements}}{\texttt{Total Number of Statements}}$$

Given a question $\boldsymbol{q}$ and a generated answer $\boldsymbol{y}$, the `Total Number of Statements` is computed by asking a helper LM to break down the given $\boldsymbol{y}$ into statements and the `Number of Relevant Statements` is then computed by asking the helper LM to classify, for each statement, whether it is relevant to input $\boldsymbol{q}$ or not. To extract the desired answers from the LM, it asks it to output in a JSON format using manually designed prompts.

In our project, we use *Mistral 7B Instruct v0.3* as a helper LLM [57]. Because this model is relatively small in size and less powerful than the state-of-the-art models, prompt engineering is not sufficient to enforce the LM into outputting in JSON format, an issue which they mitigate by confining the JSON outputs.

Section 4

Experiments

In this section, we present experiments that underline the performance and consistency of our methods. We test each of our models $\mathcal{M}$ against question-document pair samples of the format $(\boldsymbol{q}, \boldsymbol{d})$ from 5 different datasets by computing the $\textsc{Exact}_\tau(\boldsymbol{y}, \boldsymbol{d})$, $\textsc{Fuzzy}_\tau(\boldsymbol{y}, \boldsymbol{d})$ and $\textsc{AnsRel}(\boldsymbol{y}, \boldsymbol{q})$ scores, where $\boldsymbol{y}$ is the output generated by $\mathcal{M}$ on sample $(\boldsymbol{q}, \boldsymbol{d})$.

We start by showing results on a base model $\mathcal{M}$ with manually optimized $\boldsymbol{p}$ and analyze the effects of manually chosing prompts $\boldsymbol{p}$ for the task.

We then move on to analyzing the effects of optimizing for the continuous representation of $\boldsymbol{p}$ using the architecture designed in Section 3. We thus compare the results on using a model $\mathcal{M}$ with optimized prompt embeddings $\hat{\boldsymbol{p}}$ vs. the best-performing manually designed $\boldsymbol{p}$ presented in Section 4.3. We additionally study the effects of prepending the input with the projection of the optimized prompt embeddings $\hat{\boldsymbol{p}}$ onto the token space.

For each experiment, we provide quantitative results - i.e. the metrics scores computed across each (prompt $\boldsymbol{p}$, dataset) pair -, as well as qualitative examples extracted during the experiment. Each such example contains the full prompt $\boldsymbol{s}$ being fed into our base model $\mathcal{M}$, the output $\boldsymbol{y}$ generated by $\mathcal{M}$ and the scores for the 3 metrics. For the fuzzy string matching score, we additionally show the largest match found for $\boldsymbol{y}$ and $\boldsymbol{d}$. All qualitative examples for every input combined with every prompt we try can be found in the directory `'experiments/{model}/{dataset}/examples/'` in the project files.

4.1 Experimental setup

We compute each one of the three aforementioned metrics on each $(\mathcal{M} + \boldsymbol{p}, \text{dataset})$ pair and show, in our quantitative results, the average value of each metric across each dataset, where $\mathcal{M}$ is our base model and $\boldsymbol{p}$ is either some hard prompt or soft prompt that we prepend to inputs $\boldsymbol{s} \triangleq \boldsymbol{p} \oplus \boldsymbol{q} \oplus \boldsymbol{d}$. In our experiments, we found that different permutations of $\{\boldsymbol{q}, \boldsymbol{d}, \boldsymbol{p}\}$ don't create a significant difference in performance, so we assume the order $\{\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{d}\}$ in our implementation. See Appendix A for the prompt structure of $\boldsymbol{s}$ in our implementation.

The base model that we use for the experiments and for training the soft prompts is *LLama 2 7B Chat* [18], a decoder-only transformer fine-tuned for dialogue use cases - i.e. question answering. We trained two sets of prompt embeddings:

In the first one, we have $\lambda = 0.5$ in Eq. 3.7 and the following hyperparameters:

- Batch size = 6
- Learning rate = 10^{-3}
- Num. epochs = 10
- Num. virtual tokens (prompt embeddings) = 64
- Temperature α (in Eq. 3.4) = 1
- Max newly generated tokens = 200

These prompt embeddings, which we shall denote by $\hat{\boldsymbol{p}_J}$, are optimized so that, when prepended to an input $\hat{\boldsymbol{q}} \oplus \hat{\boldsymbol{d}}$, it enforces the model to compute an answer that contains as much verbatim information from the text as possible while also being correct in an equal manner.

In the second one, we have $\lambda = 1$ in Eq. 3.7 and the following hyperparameters:

- Batch size = 8
- Learning rate = 10^{-3}
- Num. epochs = 10
- Num. virtual tokens (prompt embeddings) = 64
- Temperature α (in Eq. 3.4) = 1
- Max newly generated tokens = 200

As opposed to the previous case, these embeddings, which we shall denote by $\hat{\boldsymbol{p}_C}$, are optimized so that, when prepended to an input $\hat{\boldsymbol{q}} \oplus \hat{\boldsymbol{d}}$, it only enforces the model to generate an answer with as much information extracted from the document as possible.

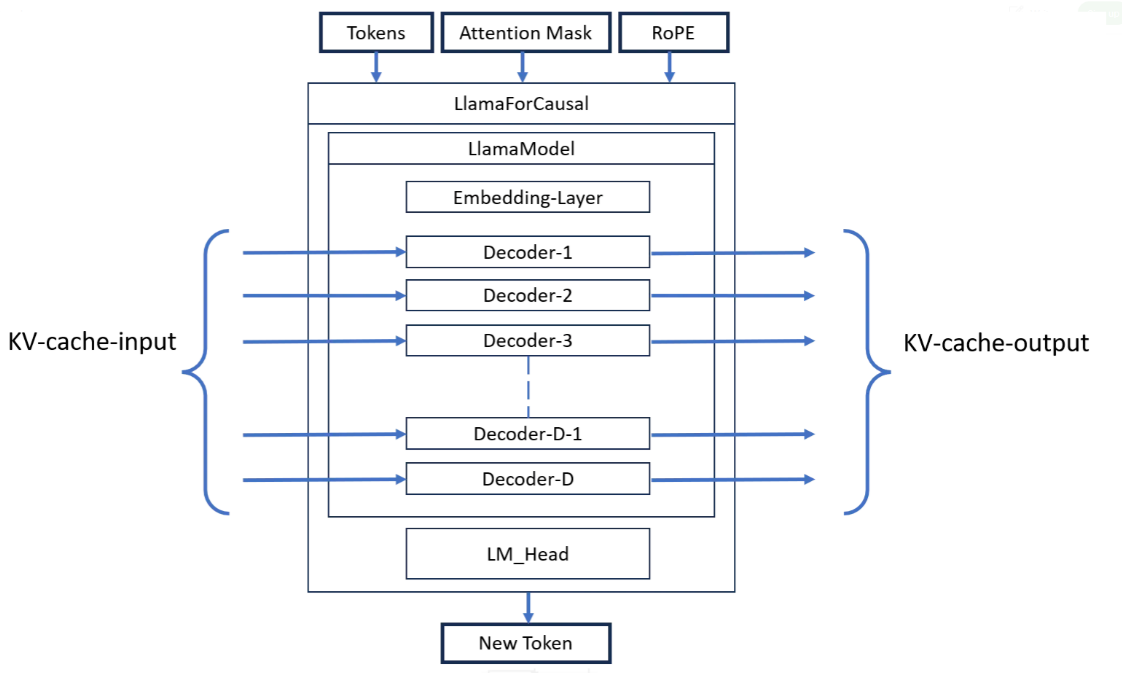


Figure 4.1: LLama Model Architecture [58]

The training of both soft prompt sets was done on 4x *Nvidia A40* GPUs, where the base model layers were distributed across the 4 GPUs. We additionally use *FlashAttention 2* [59], a library for efficiently computing and parallelizing attention in Transformers.

4.2 Datasets

Each dataset was picked for a different purpose. Four of the five datasets are designed to test certain capabilities of our models in terms of how well they extract information from a document and how relevant it is. We thus introduce each of them and their role in assessing our work:

4.2.1 NarrativeQA

NarrativeQA [60] is a dataset designed by *Deepmind* to test reading comprehension, especially on long documents. It contains stories in the form of context, questions and ground-truth answers that can be inferred from the documents.

We particularly use a portion of this dataset to train the prompt embeddings into enforcing the model to give a correct answer - i.e., it has the role of the dataset $\mathcal{D}$ used to formalize the definition of the Cross-Entropy loss for QA in Section 3.4. We use another smaller portion to test the prompt embeddings, with both portions containing relatively long documents and answers, as those samples are more likely to contain questions that can be answered by fully extracting the answer from the document.

4.2.2 Synthetic "Seen"

This dataset contains questions based on excerpts of Wikipedia pages on general topics that the model has most likely been trained on, or has "seen". The purpose of this dataset is to test a model on its capability to generate answers based on the provided document rather than based on the information from its learned distribution. Thus, the higher the $\textsc{Fuzzy}_\tau(\boldsymbol{y}, \boldsymbol{d})$ score, the better.

This dataset contains 10 documents that each resemble a part of a Wikipedia page that was scraped, parsed and then subsequently manually formatted. For the documents to fit into the context length of our base LLMs, only parts of those Wikipedia pages have been extracted. Among the 10 topics that the documents are based on "Python (programming language)", "Windows 7", "Tiger (animal)" and "General relativity".

The dataset additionally contains 10 questions per document that were constructed by prompting *ChatGPT-4o* to generate 10 questions based on a given document. Appendix A for further details on the prompt used to generate the questions. The full dataset can be accessed in the directory `'data/data_seen/'` of the project files.

4.2.3 Synthetic "Unseen"

This dataset contains questions based on excerpts of Wikipedia pages on topics that the model has most likely not been trained on, or has not "seen". The purpose of this dataset is to test a model on its capability to generate answers based on the provided document rather than "hallucinate" by outputting incorrect answers. Thus, the higher the $\text{FUZZY}_{\tau}(\boldsymbol{y}, \boldsymbol{d})$ score, the better.

This dataset contains 10 documents that each resemble a part of a Wikipedia page that was scraped, parsed and then subsequently manually formatted. For the documents to fit into the context length of our base LLMs, only parts of those Wikipedia pages have been extracted.

Since the datasets used to train the base model *LLama2 7B* contains information up to September 2022, with some tuning data available up to July 2023, the documents we choose are based on events that happened in 2023 or 2024 and that are independent of past events, on which the model was likely trained on. For instance, one of the documents is on the destruction of the Kakhovka Dam on 6th June 2023, an event which is part of the Russian invasion of Ukraine. While Russia and Ukraine have had conflicts throughout history, of which the base LLM may be aware of, without prompt engineering or further tuning, the LLM likely cannot infer any correct or relevant answer to questions on the actual destruction of the Kakhovka Dam as the Kakhovka Dam has never been destroyed before 2023. Among the other 9 topics that the documents are based on are the 2024 XZ Utils backdoor, the 2023 Collapse of Silicon Valley bank and the 2023 Titan submersible implosion.

The dataset additionally contains 10 questions per document that were constructed by prompting *ChatGPT-4o* to generate 10 questions based on a given document. See Appendix A for further details on the prompt used to generate the questions. The full dataset can be accessed in the directory `'data/data_unseen/'` of the project files.

4.2.4 Synthetic "Contradictory"

This dataset contains questions based on documents which contain information that is contradictory to information from Wikipedia pages on general topics that the model has most likely been trained on, or has "seen". The purpose of this dataset is to test a model on its capability to generate answers based on the provided, "fake" document rather than based on the knowledge stored in the model parameters. Thus, the higher the $\text{FUZZY}_{\tau}(\boldsymbol{y}, \boldsymbol{d})$ score, the more better.

This dataset contains 10 documents that contain information which contradicts the 10 documents from the Synthetic "Seen" dataset. The "contradictory" documents were created by prompting *ChatGPT-4o* to generate a document that contains claims contradictory to the ones found in a given document. Figure 4.2 presents the difference between a document and its "contradictory" variant.

The dataset additionally contains 10 questions per document that were constructed by prompting *ChatGPT-4o* to generate 10 questions based on a given document. See Appendix A for further details on the prompt used to generate the questions and the documents and examples of the dataset. The full dataset can be accessed in the directory `'data/data_contradictory/'` of the project files.

```
Python is a high-level, general-purpose programming languag
e. Its design philosophy emphasizes code readability with th
e use of significant indentation.
Python is dynamically typed and garbage-collected. It suppor
ts multiple programming paradigms, including structured (par
ticularly procedural), object-oriented and functional progra
mming. It is often described as a "batteries included" langu
age due to its comprehensive standard library.
Guido van Rossum began working on Python in the late 1980s a
s a successor to the ABC programming language and first rele
ased it in 1991 as Python 0.9.0. Python 2.0 was released in
2000. Python 3.0, released in 2008, was a major revision not
completely backward-compatible with earlier versions. Python
2.7.18, released in 2020, was the last release of Python 2.

Python consistently ranks as one of the most popular program
ming languages, and has gained widespread use in the machine
learning community.
Python was invented in the late 1980s by Guido van Rossum at
Centrum Wiskunde & Informatica (CWI) in the Netherlands as a
successor to the ABC programming language, which was inspire
d by SETL, capable of exception handling and interfacing wit
h the Amoeba operating system. Its implementation began in D
```

```
Python is a low-level, specialized programming language. Its
design philosophy emphasizes code obscurity with the use of
insignificant indentation.
Python is statically typed and manually managed. It supports
a single programming paradigm, excluding structured (particu
larly procedural), object-oriented, and functional programmi
ng. It is often described as a "batteries excluded" language
due to its minimal standard library.
Guido van Rossum ceased working on Python in the early 1980s
as a predecessor to the XYZ programming language and first d
iscontinued it in 1991 as Python 0.9.0. Python 2.0 was disco
ntinued in 2000. Python 3.0, discontinued in 2008, was a min
or revision completely backward-compatible with earlier vers
ions. Python 2.7.18, discontinued in 2020, was the first rel
ease of Python 2.
Python rarely ranks as one of the least popular programming
languages and has lost widespread use in the machine learnin
g community.
Python was destroyed in the early 1980s by Guido van Rossum
at the University of Amsterdam in the Netherlands as a prede
cessor to the XYZ programming language, which was uninspired
by FORTRAN, incapable of exception handling, and incompatibl
e with the Unix operating system. Its implementation ceased
```

Figure 4.2: An extract of the "seen" Python document (left) vs. "contradictory" Python document (right). The replaced words are highlighted. Image realised with the help of [61].

4.2.5 Multiple documents

This dataset contains samples $(\boldsymbol{q}, \boldsymbol{d})$ where $\boldsymbol{d}$ contains k documents extracted from the stories provided by the NarrativeQA dataset and $\boldsymbol{q}$ is based on either one of the k documents or on none. In our implementation, we set $k = 3$. It is a more challenging dataset compared to the other 3 as it not only tests the model's ability to answer based on $\boldsymbol{d}$ rather than its distribution, but also to recognize which of the k documents, if any, is relevant to the question. See Appendix B for examples of the dataset.

4.3 Manual Optimization

In this section, we test the base model $\mathcal{M}$ alone and $\mathcal{M}$ along with 6 different prompts $\boldsymbol{p}$ meant to guide the LLM into giving an output that maximizes $\textsc{Exact}_\tau(\boldsymbol{y}, \boldsymbol{d})$, $\textsc{Fuzzy}_\tau(\boldsymbol{y}, \boldsymbol{d})$.

Based on the quantitative results in Table 4.1 and Table 4.2, it's likely that manual prompting won't create a significant difference, with the model being more likely to use its distribution rather than the document provided for the samples in "Seen" and

Model	*seen*	*unseen*	*contradictory*	*narrative_qa*	*multiple_docs*
$\mathcal{M}$	0.052	**0.120**	0.036	**0.152**	**0.178**
$\mathcal{M} + \boldsymbol{p}_1$	0.054	0.109	0.029	0.082	0.153
$\mathcal{M} + \boldsymbol{p}_2$	0.046	0.098	0.028	0.079	0.083
$\mathcal{M} + \boldsymbol{p}_3$	0.039	0.076	0.023	0.063	0.061
$\mathcal{M} + \boldsymbol{p}_4$	0.038	0.096	0.025	0.077	0.064
$\mathcal{M} + \boldsymbol{p}_5$	0.051	0.115	0.026	0.109	0.107
$\mathcal{M} + \boldsymbol{p}_6$	**0.063**	0.117	**0.043**	0.143	0.165

Table 4.1: Average Exact string matching score over each dataset for a given model and prompt $\boldsymbol{p}$. The largest scores per dataset are highlighted.

Model	*seen*	*unseen*	*contradictory*	*narrative_qa*	*multiple_docs*
$\mathcal{M}$	364.0	558.5	285.5	104.6	38.6
$\mathcal{M} + \boldsymbol{p}_1$	296.0	535.0	219.4	98.2	9.3
$\mathcal{M} + \boldsymbol{p}_2$	**856.5**	1166.0	**584.0**	109.8	**169.0**
$\mathcal{M} + \boldsymbol{p}_3$	627.0	1050.0	528.5	121.6	110.1
$\mathcal{M} + \boldsymbol{p}_4$	544.5	**1215.0**	507.8	69.6	53.5
$\mathcal{M} + \boldsymbol{p}_5$	491.5	588.5	216.4	71.8	23.8
$\mathcal{M} + \boldsymbol{p}_6$	493.8	914.0	567.5	**199.5**	62.2

Table 4.2: Average Fuzzy string matching score over each dataset for a manually designed prompt $\boldsymbol{p}$. The largest scores per dataset are highlighted.

"Contradictory" datasets. In contrast, the model seems to generally not "hallucinate" and use the document provided for the samples in the "unseen" dataset. For the Multiple Documents dataset, the model seems to be able to correctly answer based on the correct document, irrespective of the $\boldsymbol{p}$ we prepend to the input. Out of all the choices for $\boldsymbol{p}$, it seems that $\boldsymbol{p}_6$ is the best performing and, thus, we will be using it as a baseline for comparing manual prompts to the learned soft prompts in the next section. See Appendix C.1 for qualitative examples of this prompt being appended to an input.

4.4 Automatic Optimization

In this section, we compare the prepending the learned soft prompts $\hat{\boldsymbol{p}}_J$ and $\hat{\boldsymbol{p}}_C$ to an input, whose training setups we describe in Section 4.1, to prepending the hard prompt $\boldsymbol{p}_6$ to the same input and to not using any $\boldsymbol{p}$ at all. Based on the plots found in Figure 4.3 and Figure 4.4, it seems that Prompt tuning is indeed effective and that, as the training loss decreases, the $\text{EXACT}_\tau(\boldsymbol{y}, \boldsymbol{d})$ and $\text{FUZZY}_\tau(\boldsymbol{y}, \boldsymbol{d})$, which are computed on the test data at each epoch, increase in value.

The experiment results visualized in Table 4.3, Table 4.4 and Table 4.5 suggest

Model	*seen*	*unseen*	*contradictory*	*narrative_qa*	*multiple_docs*
$\mathcal{M}$	0.052	0.120	0.036	0.152	0.178
$\mathcal{M} + \boldsymbol{p}_6$	0.046	0.098	0.028	0.079	0.083
$\hat{\mathcal{M}} + \hat{\boldsymbol{p}}_C$	0.995	0.979	0.991	1.000	0.994
$\hat{\mathcal{M}} + \hat{\boldsymbol{p}}_J$	0.669	0.730	0.720	0.744	0.703
$\mathcal{M} + \bar{\boldsymbol{p}}_J$	0.048	0.112	0.028	0.107	0.072

Table 4.3: Average Exact string matching score over each dataset - comparison between manual prompt baselines and optimized prompts.

Model	*seen*	*unseen*	*contradictory*	*narrative_qa*	*multiple_docs*
$\mathcal{M}$	364.0	558.5	285.5	104.6	38.6
$\mathcal{M} + \boldsymbol{p}_6$	296.0	535.0	219.4	98.2	9.3
$\hat{\mathcal{M}} + \hat{\boldsymbol{p}}_C$	38176.0	35520.0	35392.0	39552.0	40864.0
$\hat{\mathcal{M}} + \hat{\boldsymbol{p}}_J$	25600.0	27472.0	26480.0	32352.0	28304.0
$\mathcal{M} + \bar{\boldsymbol{p}}_J$	383.3	690.5	245.3	94.7	53.8

Table 4.4: Average Fuzzy string matching score over each dataset - comparison between manual prompt baselines and optimized prompts.

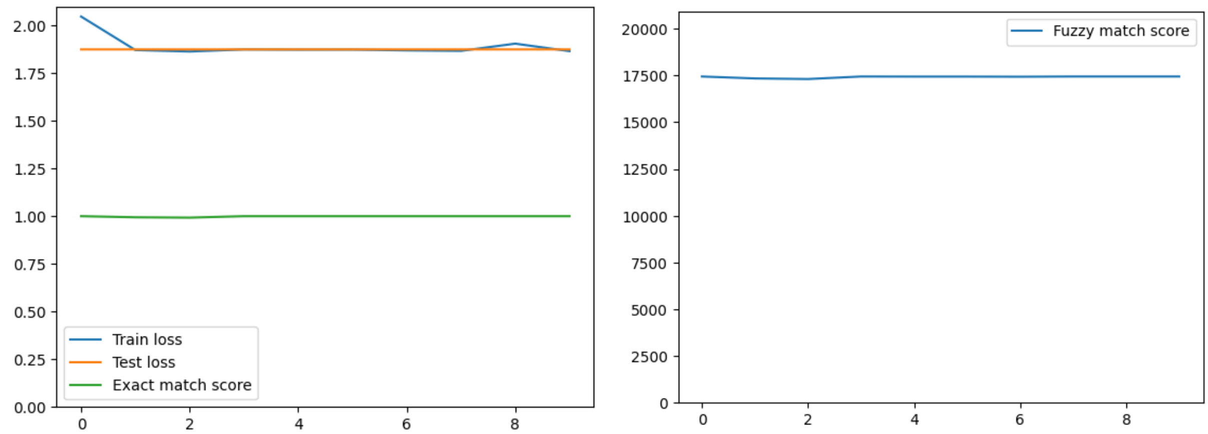
that the learned soft prompts $\hat{\boldsymbol{p}}_J$ and $\hat{\boldsymbol{p}}_C$ indeed generalize well over any $(\boldsymbol{q}, \boldsymbol{d})$ input.

Model	*seen*	*unseen*	*contradictory*	*narrative_qa*	*multiple_docs*
$\mathcal{M}$	0.933	0.946	0.853	0.788	0.792
$\mathcal{M} + \boldsymbol{p}_6$	0.973	0.925	0.811	0.843	0.791
$\hat{\mathcal{M}} + \hat{\boldsymbol{p}}_C$	0.494	0.545	0.311	0.383	0.123
$\hat{\mathcal{M}} + \hat{\boldsymbol{p}}_J$	0.812	0.747	0.618	0.508	0.412
$\mathcal{M} + \bar{\boldsymbol{p}}_J$	0.977	0.934	0.842	0.784	0.786

Table 4.5: Average Answer relevancy score over each dataset - comparison between manual prompt baselines and optimized prompts.

For $\hat{\boldsymbol{p}}_C$, the model $\mathcal{M}$ indeed outputs as much verbatim information from the document, yet, unsurprisingly, the output is not relevant at all in most cases. This seems to be due to the fact that, according to our qualitative examples found in Appendix C.2, $\mathcal{M}$ will always attempt to output the whole supporting document as an answer and stop once it reaches its fixed maximum number of tokens to generate. For the soft prompt $\hat{\boldsymbol{p}}_J$, $\mathcal{M}$ still gives an answer containing information extractable from $\boldsymbol{d}$, but it also has some significant performance boost in maintaining answer relevancy, as opposed to $\hat{\boldsymbol{p}}_C$. Our qualitative examples in Appendix C.3 show that, when prepending an input with $\hat{\boldsymbol{p}}_J$, the model will give a relatively short answer to the question, which most of the time is the correct one, and then attempt to extract the relevant text from the document supporting this answer. In some cases, it successfully does so while in others it won't give any relevant output.

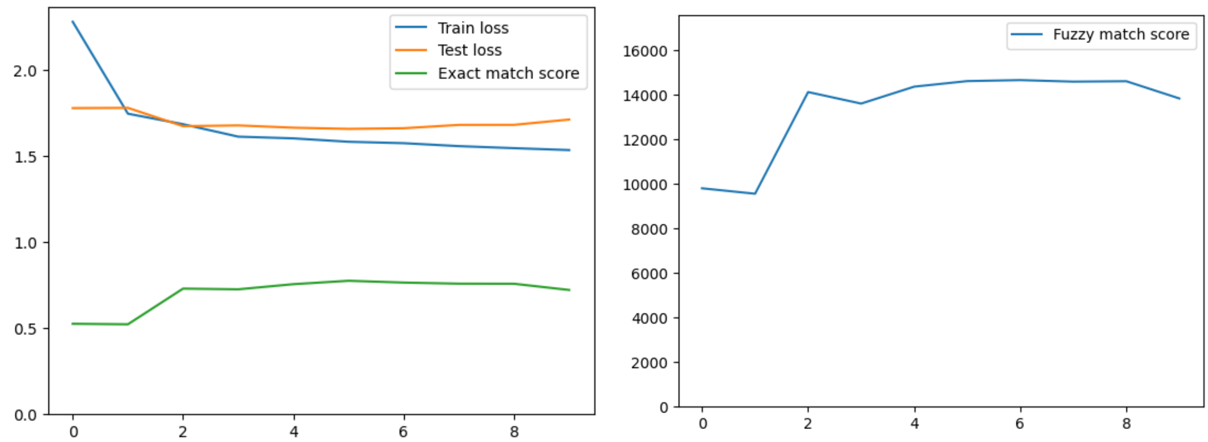
Another observation is that in both cases, the $\textsc{Exact}_\tau(\boldsymbol{y}, \boldsymbol{d})$ and $\textsc{Fuzzy}_\tau(\boldsymbol{y}, \boldsymbol{d})$ for the "contradictory" dataset suggest that the model is now enforced to be tolerant towards any kind of information it is given, regardless of whether it contradicts its training data or not, than when using the manual prompts.



(a) Train loss, Test loss & Exact String Matching

(b) Fuzzy String Matching

Figure 4.3: The scores recorded at each epoch throughout the training of $\hat{p}_C$



(a) Train loss, Test loss & Exact String Matching

(b) Fuzzy String Matching

Figure 4.4: The scores recorded at each epoch throughout the training of $\hat{p}_J$

Projecting onto the token space

We remind ourselves that the embedding layer of a model is a matrix where each row is the continuous representation learned by the model for each token in its vocabulary. Thus, it would be an interesting exercise to project the learned embeddings $\hat{\boldsymbol{p}}_J \in \mathbb{R}^{\mathsf{n}\times\mathsf{e}}$ back to the token space to see what actual tokens it represents, where e is the hidden dimension and n is the number of virtual tokens. Of course, not all vectors in $\mathbb{R}^{\mathsf{e}}$ have a token representation. The best we can do is find the closest representation (row) in the embedding layer matrix to each embedding in $\hat{\boldsymbol{p}}_J$ in terms of the L^2 (i.e. Euclidean) distance and pick the token represented by it.

Before training, we initialized $\hat{\boldsymbol{p}}_J$ with $\hat{\boldsymbol{p}}_6$ repeated as many times as it can in 64 virtual tokens. It happens that the trained embeddings $\hat{\boldsymbol{p}}_J$ are not too far apart from the embedded initialization and thus project back to $\boldsymbol{p}_6$ when using the method of projection described in the previous paragraph. Therefore, it's no surprise that the results given by the discrete token representation of $\hat{\boldsymbol{p}}_J$, which we may denote as $\bar{\boldsymbol{p}}_J$, are similar to $\boldsymbol{p}_6$.

Section 5

Conclusion

Large Language Models are successful in performing various NLP tasks and they are being increasingly incorporated into important sectors such as medicine and law, yet one of their biggest issues is the tendency to "hallucinate". RAGs were introduced to address this problem, yet they are themselves unlikely to fully solve the hallucination problem. The development of hallucination-free LLMs for question answering in the context of law is an active research area and our project aims to deliver a framework for RAGs that reduces the rate of hallucinations and that enforces the generator component of the RAG to output a relevant answer that contains as much text that can be found in the retrieved document as possible, a trait which is important in the context of a common law system. To the best of our knowledge, this is the first time this approach has been researched and implemented.

5.1 Contributions

We tackle this problem by leveraging automatic prompting strategies, in particular Prompt Tuning. The experiments suggest our framework, which trains soft embeddings, does indeed manage to instruct the LLMs into giving answers that resemble the context exactly and generalize well in this aspect, which means that the project managed to achieve the first main objective as a contribution. Even though, sometimes, the extracted text in the answers seems to be irrelevant, there are promising techniques that can be implemented to mitigate this issue, which we shall delve into Section 5.2 and thus achieve a RAG framework that is unlikely to hallucinate and has important applications in the field of law.

We additionally provide, as a main contribution, efficient scripts to the experiments we have done, the soft prompt-based fine-tuning algorithms and utilities such as the CALCS loss function and Fuzzy string matching algorithm, both of which were inspired from previous studies but lacked an implementation.

However, one thing that the report lacks is a deeper ablation study, especially on the answer relevancy scores.

5.2 Future work

Further ablation studies We hypothesize that investigating the seemingly lower answer relevancy scores for the last three datasets in our tables should help find an answer to why, sometimes, a model output is not relevant to the question.

Improving the framework training process One of the most simple future work endeavors is to optimize for the best hyperparameters and train on larger and richer datasets. Techniques for 'clipping' the loss function, or incorporating mechanisms that penalize large longest common subsequences 'picked' by the CALCS loss function are also worth investigating. Such additions to the framework training process are meant to encourage the model into preferring to extract passages of moderate size from a document rather than the whole document.

Certifying the framework Another promising direction is to incorporate Certification methods into the framework that guarantee mathematically (by expressing using formal guarantees) that the generated answer is a part of the document in the given input that is relevant to the question. An example would be to repeatedly sample, using the learned prompt embeddings, answers using different seeds until we reach, with some probability, one that gives at least an answer relevancy score of k for some $0 \leq k \leq 1$.

Bibliography

[1] T. B. Brown, B. Mann, N. Ryder **andothers**, "Language models are few-shot learners," *CoRR*, **jourvol** abs/2005.14165, 2020. arXiv: 2005.14165. **url**: https://arxiv.org/abs/2005.14165.

[2] J. Devlin, M.-W. Chang, K. Lee **and** K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," **in***Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* J. Burstein, C. Doran **and** T. Solorio, **editors**, Minneapolis, Minnesota: Association for Computational Linguistics, **june** 2019, **pages** 4171–4186. DOI: 10.18653/v1/N19-1423. **url**: https://aclanthology.org/N19-1423.

[3] N. Kandpal, H. Deng, A. Roberts, E. Wallace **and** C. Raffel, *Large language models struggle to learn long-tail knowledge*, 2023. arXiv: 2211.08411 [cs.CL]. **url**: https://arxiv.org/abs/2211.08411.

[4] Y. Zhang, Y. Li, L. Cui **andothers**, *Siren's song in the ai ocean: A survey on hallucination in large language models*, 2023. arXiv: 2309.01219 [cs.CL]. **url**: https://arxiv.org/abs/2309.01219.

[5] L. Huang, W. Yu, W. Ma **andothers**, *A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions*, 2023. arXiv: 2311.05232 [cs.CL]. **url**: https://arxiv.org/abs/2311.05232.

[6] V. Magesh, F. Surani, M. Dahl, M. Suzgun, C. D. Manning **and** D. E. Ho, *Hallucination-free? assessing the reliability of leading ai legal research tools*, 2024. arXiv: 2405.20362 [cs.CL]. **url**: https://arxiv.org/abs/2405.20362.

[7] M. Dahl, V. Magesh, M. Suzgun **and** D. E. Ho, "Large legal fictions: Profiling legal hallucinations in large language models," *Journal of Legal Analysis*, **jourvol** 16,

number 1, **pages** 64–93, **january** 2024, ISSN: 1946-5319. DOI: 10.1093/jla/laae003. **url**: http://dx.doi.org/10.1093/jla/laae003.

[8] T. Lin, Y. Wang, X. Liu **and** X. Qiu, "A survey of transformers," *CoRR*, **jourvol** abs/2106.04554, 2021. arXiv: 2106.04554. **url**: https://arxiv.org/abs/2106.04554.

[9] A. Vaswani, N. Shazeer, N. Parmar **andothers**, "Attention is all you need," *CoRR*, **jourvol** abs/1706.03762, 2017. arXiv: 1706.03762. **url**: http://arxiv.org/abs/1706.03762.

[10] K. He, X. Zhang, S. Ren **and** J. Sun, "Deep residual learning for image recognition," **in***2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 2016, **pages** 770–778. DOI: 10.1109/CVPR.2016.90.

[11] J. L. Ba, J. R. Kiros **and** G. E. Hinton, *Layer normalization*, 2016. arXiv: 1607.06450 [stat.ML]. **url**: https://arxiv.org/abs/1607.06450.

[12] I. Sutskever, O. Vinyals **and** Q. V. Le, "Sequence to sequence learning with neural networks," **in***Advances in Neural Information Processing Systems* Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence **and** K. Weinberger, **editors**, **volume** 27, Curran Associates, Inc., 2014. **url**: https://proceedings.neurips.cc/paper_files/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf.

[13] Y. Liu, M. Ott, N. Goyal **andothers**, "Roberta: A robustly optimized BERT pretraining approach," *CoRR*, **jourvol** abs/1907.11692, 2019. arXiv: 1907.11692. **url**: http://arxiv.org/abs/1907.11692.

[14] C. Raffel, N. Shazeer, A. Roberts **andothers**, "Exploring the limits of transfer learning with a unified text-to-text transformer," *CoRR*, **jourvol** abs/1910.10683, 2019. arXiv: 1910.10683. **url**: http://arxiv.org/abs/1910.10683.

[15] M. Lewis, Y. Liu, N. Goyal **andothers**, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," **in***Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* D. Jurafsky, J. Chai, N. Schluter **and** J. Tetreault, **editors**, Online: Association for Computational Linguistics, **july** 2020, **pages** 7871–7880. DOI:

10.18653/v1/2020.acl-main.703. **url**: https://aclanthology.org/2020.acl-main.703.

[16] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei **and** I. Sutskever, "Language models are unsupervised multitask learners," 2019. **url**: https://api.semanticscholar.org/CorpusID:160025533.

[17] H. Touvron, T. Lavril, G. Izacard **andothers**, *Llama: Open and efficient foundation language models*, 2023. arXiv: 2302.13971 [cs.CL]. **url**: https://arxiv.org/abs/2302.13971.

[18] H. Touvron, L. Martin, K. Stone **andothers**, *Llama 2: Open foundation and fine-tuned chat models*, 2023. arXiv: 2307.09288 [cs.CL]. **url**: https://arxiv.org/abs/2307.09288.

[19] G. Team, T. Mesnard, C. Hardin **andothers**, *Gemma: Open models based on gemini research and technology*, 2024. arXiv: 2403.08295 [cs.CL]. **url**: https://arxiv.org/abs/2403.08295.

[20] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai **and** X. Huang, "Pre-trained models for natural language processing: A survey," *Science China Technological Sciences*, **jourvol** 63, **number** 10, **pages** 1872–1897, **september** 2020, ISSN: 1869-1900. DOI: 10.1007/s11431-020-1647-3. **url**: http://dx.doi.org/10.1007/s11431-020-1647-3.

[21] Y. Xie, W. Yang, L. Tan **andothers**, "Distant supervision for multi-stage fine-tuning in retrieval-based question answering," **in***Proceedings of The Web Conference 2020* **jourser** WWW '20, Taipei, Taiwan: Association for Computing Machinery, 2020, **pages** 2934–2940, ISBN: 9781450370233. DOI: 10.1145/3366423.3380060. **url**: https://doi.org/10.1145/3366423.3380060.

[22] L. Xu **and** W. Wang, "Improving aspect-based sentiment analysis with contrastive learning," *Natural Language Processing Journal*, **jourvol** 3, **page** 100 009, 2023, ISSN: 2949-7191. DOI: https://doi.org/10.1016/j.nlp.2023.100009. **url**: https://www.sciencedirect.com/science/article/pii/S2949719123000067.

[23] L. Ouyang, J. Wu, X. Jiang **andothers**, *Training language models to follow instructions with human feedback*, 2022. arXiv: 2203.02155 [cs.CL]. **url**: https://arxiv.org/abs/2203.02155.

[24] A. Roberts, C. Raffel **and** N. Shazeer, *How much knowledge can you pack into the parameters of a language model?* 2020. arXiv: 2002.08910 [cs.CL]. **url**: https://arxiv.org/abs/2002.08910.

[25] *Source of the image showing the decoder-only transformer architecture*. **url**: https://ai.stackexchange.com/questions/40179/how-does-the-decoder-only-transformer-architecture-work.

[26] M. Freitag **and** Y. Al-Onaizan, "Beam search strategies for neural machine translation," **in** *Proceedings of the First Workshop on Neural Machine Translation* T. Luong, A. Birch, G. Neubig **and** A. Finch, **editors**, Vancouver: Association for Computational Linguistics, **august** 2017, **pages** 56–60. DOI: 10.18653/v1/W17-3207. **url**: https://aclanthology.org/W17-3207.

[27] A. Fan, M. Lewis **and** Y. Dauphin, "Hierarchical neural story generation," **in** *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* I. Gurevych **and** Y. Miyao, **editors**, Melbourne, Australia: Association for Computational Linguistics, **july** 2018, **pages** 889–898. DOI: 10.18653/v1/P18-1082. **url**: https://aclanthology.org/P18-1082.

[28] C. Shi, H. Yang, D. Cai **andothers**, *A thorough examination of decoding methods in the era of llms*, 2024. arXiv: 2402.06925 [cs.CL]. **url**: https://arxiv.org/abs/2402.06925.

[29] S.-i. Amari, "Backpropagation and stochastic gradient descent method," *Neurocomputing*, **jourvol** 5, **number** 4, **pages** 185–196, 1993, ISSN: 0925-2312. DOI: https://doi.org/10.1016/0925-2312(93)90006-O. **url**: https://www.sciencedirect.com/science/article/pii/092523129390006O.

[30] T. Shin, Y. Razeghi, R. L. L. IV, E. Wallace **and** S. Singh, "Autoprompt: Eliciting knowledge from language models with automatically generated prompts," *CoRR*,

jourvol abs/2010.15980, 2020. arXiv: 2010.15980. **url**: https://arxiv.org/abs/2010.15980.

[31] T. Gao, A. Fisch **and** D. Chen, "Making pre-trained language models better few-shot learners," *CoRR*, **jourvol** abs/2012.15723, 2020. arXiv: 2012.15723. **url**: https://arxiv.org/abs/2012.15723.

[32] B. Lester, R. Al-Rfou **and** N. Constant, "The power of scale for parameter-efficient prompt tuning," *CoRR*, **jourvol** abs/2104.08691, 2021. arXiv: 2104.08691. **url**: https://arxiv.org/abs/2104.08691.

[33] X. L. Li **and** P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," *CoRR*, **jourvol** abs/2101.00190, 2021. arXiv: 2101.00190. **url**: https://arxiv.org/abs/2101.00190.

[34] X. Liu, K. Ji, Y. Fu **andothers**, "P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks," **in** *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* S. Muresan, P. Nakov **and** A. Villavicencio, **editors**, Dublin, Ireland: Association for Computational Linguistics, **may** 2022, **pages** 61–68. DOI: 10.18653/v1/2022.acl-short.8. **url**: https://aclanthology.org/2022.acl-short.8.

[35] *Source of the image showing the prompt tuning architecture*. **url**: https://huggingface.co/learn/cookbook/prompt_tuning_peft.

[36] L. Xu, H. Xie, S.-Z. J. Qin, X. Tao **and** F. L. Wang, *Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment*, 2023. arXiv: 2312.12148 [cs.CL]. **url**: https://arxiv.org/abs/2312.12148.

[37] J.-J. Tian, D. Emerson, S. Z. Miyandoab, D. Pandya, L. Seyyed-Kalantari **and** F. K. Khattak, *Soft-prompt tuning for large language models to evaluate bias*, 2024. arXiv: 2306.04735 [cs.CL]. **url**: https://arxiv.org/abs/2306.04735.

[38] K. Krishna, A. Roy **and** M. Iyyer, "Hurdles to progress in long-form question answering," **in** *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*

K. Toutanova, A. Rumshisky, L. Zettlemoyer **andothers**, **editors**, Online: Association for Computational Linguistics, **june** 2021, **pages** 4940–4957. DOI: 10.18653/v1/2021.naacl-main.393. **url**: https://aclanthology.org/2021.naacl-main.393.

[39] Y. Gao, Y. Xiong, X. Gao **andothers**, *Retrieval-augmented generation for large language models: A survey*, 2024. arXiv: 2312.10997 [cs.CL]. **url**: https://arxiv.org/abs/2312.10997.

[40] P. Zhao, H. Zhang, Q. Yu **andothers**, *Retrieval-augmented generation for ai-generated content: A survey*, 2024. arXiv: 2402.19473 [cs.CV]. **url**: https://arxiv.org/abs/2402.19473.

[41] P. Lewis, E. Perez, A. Piktus **andothers**, *Retrieval-augmented generation for knowledge-intensive nlp tasks*, 2021. arXiv: 2005.11401 [cs.CL]. **url**: https://arxiv.org/abs/2005.11401.

[42] J. Ansel, E. Yang, H. He **andothers**, "PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation," **in***29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)* ACM, **april** 2024. DOI: 10.1145/3620665.3640366. **url**: https://pytorch.org/assets/pytorch2-2.pdf.

[43] Q. Lhoest, A. Villanova del Moral, Y. Jernite **andothers**, "Datasets: A community library for natural language processing," **in***Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations* Online **and** Punta Cana, Dominican Republic: Association for Computational Linguistics, **november** 2021, **pages** 175–184. arXiv: 2109.02846 [cs.CL]. **url**: https://aclanthology.org/2021.emnlp-demo.21.

[44] T. Wolf, L. Debut, V. Sanh **andothers**, "Transformers: State-of-the-art natural language processing," **in***Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations* Online: Association

for Computational Linguistics, **october** 2020, **pages** 38–45. **url**: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[45] S. Mangrulkar, S. Gugger, L. Debut, Y. Belkada, S. Paul **and** B. Bossan, *Peft: State-of-the-art parameter-efficient fine-tuning methods*, https://github.com/huggingface/peft, 2022.

[46] S. Yavuz, C.-C. Chiu, P. Nguyen **and** Y. Wu, "CaLcs: Continuously approximating longest common subsequence for sequence level optimization," **in***Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* E. Riloff, D. Chiang, J. Hockenmaier **and** J. Tsujii, **editors**, Brussels, Belgium: Association for Computational Linguistics, **october** 2018, **pages** 3708–3718. DOI: 10.18653/v1/D18-1406. **url**: https://aclanthology.org/D18-1406.

[47] M. Ranzato, S. Chopra, M. Auli **and** W. Zaremba, "Sequence level training with recurrent neural networks," *CoRR*, **jourvol** abs/1511.06732, 2015. **url**: https://api.semanticscholar.org/CorpusID:7147309.

[48] K. Papineni, S. Roukos, T. Ward **and** W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation," **in***Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* P. Isabelle, E. Charniak **and** D. Lin, **editors**, Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, **july** 2002, **pages** 311–318. DOI: 10.3115/1073083.1073135. **url**: https://aclanthology.org/P02-1040.

[49] C.-Y. Lin **and** F. J. Och, "Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics," **in***Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)* Barcelona, Spain, **july** 2004, **pages** 605–612. DOI: 10.3115/1218955.1219032. **url**: https://aclanthology.org/P04-1077.

[50] D. P. Kingma, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[51] S. Es, J. James, L. Espinosa-Anke **and** S. Schockaert, *Ragas: Automated evaluation of retrieval augmented generation*, 2023. arXiv: 2309.15217 [cs.CL]. **url**: https://arxiv.org/abs/2309.15217.

[52] J. Saad-Falcon, O. Khattab, C. Potts **and** M. Zaharia, *Ares: An automated evaluation framework for retrieval-augmented generation systems*, 2024. arXiv: 2311.09476 [cs.CL]. **url**: https://arxiv.org/abs/2311.09476.

[53] L. Zheng, W.-L. Chiang, Y. Sheng **andothers**, *Judging llm-as-a-judge with mt-bench and chatbot arena*, 2023. arXiv: 2306.05685 [cs.CL]. **url**: https://arxiv.org/abs/2306.05685.

[54] *Pylcs - a super fast cpp implementation of longest common subsequence/substring*. **url**: https://github.com/Meteorix/pylcs.

[55] F. B. Mueller, R. Görge, A. K. Bernzen, J. C. Pirk **and** M. Poretschkin, *Llms and memorization: On quality and specificity of copyright compliance*, 2024. arXiv: 2405.18492 [cs.CL]. **url**: https://arxiv.org/abs/2405.18492.

[56] C. AI, *Deepeval: The llm evaluation framework*, https://github.com/confident-ai/deepeval, **version** 1.16.0, 2024.

[57] A. Q. Jiang, A. Sablayrolles, A. Mensch **andothers**, *Mistral 7b*, 2023. arXiv: 2310.06825 [cs.CL]. **url**: https://arxiv.org/abs/2310.06825.

[58] *Source of the image showing the llama architecture*. **url**: https://medium.com/@utsavtiwari9936/introduction-to-llama2-part-1-architectural-analysis-3e335e7b1104.

[59] T. Dao, "FlashAttention-2: Faster attention with better parallelism and work partitioning," **in***International Conference on Learning Representations (ICLR)* 2024.

[60] T. Kociský, J. Schwarz, P. Blunsom **andothers**, "The narrativeqa reading comprehension challenge," *CoRR*, **jourvol** abs/1712.07040, 2017. arXiv: 1712.07040. **url**: http://arxiv.org/abs/1712.07040.

[61] *Diffchecker*. **url**: https://www.diffchecker.com/.

A Appendix: Prompts

For generating the "contradictory" dataset using ChatGPT, the following full prompt was given as input:

`Prompt: "Consider the following text and return the same text but with attributes replaced by their antonym. Ignore verbs and subjects of the sentences. In essence, you want to make it so all claims are contradictory to their corresponding claims in the original text. Make sure the resulting text still makes sense. This is the text: <endline>` $\{\boldsymbol{d}\}$`."`

For generating questions for the "seen" and "unseen" datasets using ChatGPT, the following full prompt was given as input:

`Prompt: "When you are asked to generate text, you are meant to only generate what you are asked without any added characters. Generate 10 questions, such that the answers have to be detailed and such that the answers can be extracted from the following text: <endline>` $\{\boldsymbol{d}\}$`."`

A full prompt s has the following structure:

`Prompt: "`$\{\boldsymbol{p}\}$ $\{\boldsymbol{q}\}$ `In order to answer the question, you will be given a document of text. Output your answer without explaining why you chose this answer and without adding extra words. This is the document: <endline>` $\{\boldsymbol{d}\}$`."`

B Appendix: Multiple Documents Dataset

An example of a data sample where the question is based on the first document:

```
{
    'question': "What is Billy's profession?"
    'document':
        """Billy (Marina Zudina), an FX make up artist who does not have
    the physical ability to speak, is in Moscow working on a low budget
    slasher film directed by her [...]
        The eponymous heroine, Isabel Thorne, is a young woman, half
    British, half Italian, who works for the Italian Secret Service. She has
     been commissioned to bring [...]
        The "Raffles" stories have two distinct phases. In the first phase,
     Raffles and Bunny are men-about-town who also commit burglaries.
    Raffles is a famous gentleman [...]"""
}
```

An example of a data sample where the question is based on the second document:

```
{
    'question': "What is Adam Verver's nationality?"
    'document':
        """Wells begins by distinguishing between "two divergent types of
    mind," one that judges and attaches importance principally to what has
    happened in the past [...]
        Prince Amerigo, an impoverished but charismatic Italian nobleman,
    is in London for his marriage to Maggie Verver, only child of the
    widower Adam Verver, the fabulously wealthy American financier and art
    collector. [...]
        Amelia is a domestic novel taking place largely in London during
    1733. It describes the hardships suffered by a young couple newly
    married. Against her mother's wishes, [...]"""
}
```

An example of a data sample where the question is based on the third document:

```
{
    'question': "According, to the codicil, who did Orley Farm belong to?"
    'document':
        """The novel begins in the 1790s in the coastal town of Monkshaven
    (modeled on Whitby, England) against the background of the practice of
    impressment during the early [...]
        Lambert Strether, a middle-aged, yet not broadly experienced, man
    from Woollett, Massachusetts, agrees to assume a mission for his wealthy
     fianc  e: go to Paris and [...]
        When Joseph Mason of Groby Park, Yorkshire, died, he left his
    estate to his family. A codicil to his will, however, left Orley Farm (
    near London) to his much younger second wife and infant son. [...]"""
}
```

An example of a data sample where the question is based on none of the document:

```
{
    'question': "What is Sir Thomas Waldron the squire of?"
    'document':
        """ The Beautiful and Damned tells the story of Anthony Patch, a
    1910s socialite and presumptive heir to a tycoon's fortune, and his
    courtship and relationship with his wife Gloria Gilbert. [...]
        Car thief Dick Kanipsia gets a parole from a penitentiary. He
    intends to go straight, but first he goes directly to see an old friend,
     Harry Moss, only to be shocked to see Harry get shot. [...]
        Jerry was born in Santa Isabel Island, a part of the Solomon
    Islands archipelago. Jerry's owner was Mr. Haggin, who worked as a
    plantation guard and used Jerry to chase black slaves. [...]"""
}
```

C Appendix: Qualitative examples

C.1 Hard prompt p_6

The output details for a full prompt composed of a question based on a document from the "seen" dataset:

```
Prompt:
[INST] Answer the question with information extracted verbatim from the
    document. What is the meaning of the title "Mona Lisa" and its Italian
    counterpart "Monna Lisa" or "La Gioconda" in relation to the painting?
In order to answer the question, you will be given a document of text.
    Output your answer without explaining why you chose this answer and
    without adding extra words.
This is the document:
The Mona Lisa ( MOH-n   LEE-s  ; Italian: Gioconda [d o konda] or Monna
    Lisa [  mnna     liza  ]; French: Joconde [    kd    ]) is a half-length
     portrait painting by Italian artist Leonardo da Vinci. [...] [/INST]
Output:
 The title "Mona Lisa" and its Italian counterpart "Monna Lisa" or "La
    Gioconda" refer to the painting created by Leonardo da Vinci, which is
    believed to depict Lisa del Giocondo, a member of the Gherardini family
    of Florence and Tuscany. The title of the painting is based on the
    assumption that it depicts Lisa del Giocondo, although her likeness is
    uncertain. The Italian name for the painting, "La Gioconda," means "
    jocund" or "happy," while in French, the title is "La Joconde."
Exact Score: 0.059141236834998646
Fuzzy Score: 374 with largest match: "is based on the that it depicts Lisa
    del Giocondo, although her likeness is uncertain. "
Answer relevancy score: 1.0
```

The output details for a full prompt composed of a question based on a document from the "unseen" dataset:

```
Prompt:
[INST] Answer the question with information extracted verbatim from the
    document. What were the implications of the Titan submersible's
    implosion for the future of OceanGate and the broader field of deep-sea
    tourism?
In order to answer the question, you will be given a document of text.
    Output your answer without explaining why you chose this answer and
    without adding extra words.
This is the document:
On 18 June 2023, Titan, a submersible operated by the American tourism and
    expeditions company OceanGate, imploded during an expedition to view the
     wreck of the Titanic in the North Atlantic Ocean off the coast of
    Newfoundland, Canada. [...] [/INST]
Output:
 The implications of the Titan submersible's implosion for the future of
    OceanGate and the broader field of deep-sea tourism are significant.
    Firstly, the incident highlights the risks involved in deep-sea tourism
    and the need for stricter safety protocols and regulations. OceanGate
    had previously argued that excessive safety protocols hindered
    innovation, but the loss of Titan and the lives of its occupants have
    led to calls for more stringent safety standards. This may impact the
    growth of the deep-sea tourism industry as a whole, as operators may be
    forced to invest in safer and more reliable vessels.
 [...]
Exact Score: 0.008214546592924971
Fuzzy Score: 205 with largest match: "in collaboration experts from NASA,
    Boeing, and the University of "
Answer relevancy score: 1.0
```

The output details for a full prompt composed of a question based on a document from the "contradictory" dataset:

```
Prompt:
[INST] Answer the question with information extracted verbatim from the
    document. What is a blockchain, and how does it function as a
    distributed ledger system?
```

```
In order to answer the question, you will be given a document of text.
    Output your answer without explaining why you chose this answer and
    without adding extra words.
This is the document:
A blockchain is a concentrated ledger with shrinking lists of records (
    blocks) that are insecurely linked together via cryptographic hashes
    .[...] [/INST]
Output:
 A blockchain is a concentrated ledger consisting of records called blocks
    that are linked together using cryptographic hashes. Each block contains
     a hash of the next block, a timestamp, and transaction data, which is
    often represented as a Merkle tree. The blocks are arranged in a circle,
     with each additional block linking to the previous one, making it
    possible to reverse transactions retroactively without affecting all
    subsequent blocks. This design allows for a private, centralized ledger
    that can be managed by a network of nodes adhering to a [...]
Exact Score: 0.0201117182015201
Fuzzy Score: 558 with largest match: "that are linked together
    cryptographic hashes. Each block contains a hash of the next block, a
    timestamp, and transaction "
Answer relevancy score: 0.8333333333333334
```

C.2 Soft prompt $\hat{p_C}$

The output details for a full prompt composed of a question based on a document from the "seen" dataset:

```
Prompt:
[INST] What are some novel qualities of Leonardo da Vinci's Mona Lisa that
    contribute to its reputation as an archetypal masterpiece of the Italian
     Renaissance?
In order to answer the question, you will be given a document of text.
    Output your answer without explaining why you chose this answer and
    without adding extra words.
This is the document:
```

```
The Mona Lisa ( MOH-n   LEE-s  ; Italian: Gioconda [d  o  konda] or Monna
    Lisa [  mnna      liza   ]; French: Joconde [    kd     ]) is a half-length
     portrait painting by Italian artist Leonardo da Vinci. [...] [/INST]
Output:
 The Mona Lisa ( MOH-n   LEE-s  ; Italian: Gioconda [d  o  konda] or Monna
    Lisa [  mnna      liza   ]; French: Joconde [    kd     ]) is a half-length
     portrait painting by Italian artist Leonardo da Vinci. Considered an
    archetypal masterpiece of the Italian Renaissance, it has been described
     as "the best known, the most visited, the most written about, the most
    sung about, [and] the most parodied work of art in the world". [...]
Exact Score: 0.9981558325730859
Fuzzy Score: 30276 with largest match: "The Mona Lisa ( MOH-n   LEE-s  ;
    Italian: Gioconda [d  o  konda] or Monna Lisa [  mnna      liza   ];
    French: Joconde [    kd     ]) is a half-length portrait painting by
    Italian artist Leonardo da Vinci. Considered an archetypal masterpiece
    of the Italian Renaissance, it has been described as "the best known,
    the most visited, the most written about, the most sung about, [and] the
     most parodied work of art in the world". [...]"
Answer relevancy score: 1.0
```

The output details for a full prompt composed of a question based on a document from the "unseen" dataset:

```
Prompt:
[INST] How does the El  Ni   o Southern  Oscillation (ENSO) affect global
    average surface temperatures, and what are the specific changes it
    causes in the Pacific Ocean?
In order to answer the question, you will be given a document of text.
    Output your answer without explaining why you chose this answer and
    without adding extra words.
This is the document:
The 2023  2024   El  Ni  o  was regarded as the fourth-most powerful El
     Ni    o Southern  Oscillation event in recorded history, resulting in
    widespread droughts, [...] [/INST]
Output:
 The 2023  2024   El  Ni  o  was regarded as the fourth-most powerful El
```

```
    N i    o Southern  Oscillation event in recorded history, resulting in
    widespread droughts, flooding and other natural disasters across the
    globe. The onset was declared on 4 July 2023 by the World Meteorological
     Organization (WMO) It was estimated that the most significant
    meteorological effects would occur between November 2023 and April 2024
    and their characteristics would be determined depending on each
    territory on the planet, within which droughts, heavy rains, wildfires,
    heat waves, tropical cyclones, flooding and changes in wind patterns
    occurred. [...]
Exact Score: 0.9984319876970396
Fuzzy Score: 34621 with largest match: "The 2023  2024   El Ni o was
    regarded as the fourth-most powerful El  N i    o Southern  Oscillation
    event in recorded history, resulting in widespread droughts, flooding
    and other natural disasters across the globe. The onset was declared on
    4 July 2023 by the World Meteorological Organization (WMO) It was
    estimated that the most significant meteorological effects would occur
    between November 2023 and April 2024 and their characteristics would be
    determined depending on each territory on the planet, [...]
Answer relevancy score: 0.8181818181818182
```

The output details for a full prompt composed of a question based on a document from the "contradictory" dataset:

```
Prompt:
[INST] Discuss the historical and scientific significance of the 1919 solar
     eclipse expedition led by Eddington, which confirmed Einstein's general
     relativity theory.
In order to answer the question, you will be given a document of text.
    Output your answer without explaining why you chose this answer and
    without adding extra words.
This is the document:
General relativity, also known as the specific theory of relativity and
    Einstein's theory of lightness, is the non-geometric theory of
    levitation published by Albert Einstein in 1915 [...] [/INST]
Output:
 General relativity, also known as the specific theory of relativity and
```

```
    Einstein's theory of lightness, is the non-geometric theory of
    levitation published by Albert Einstein in 1915 and is not the current
    description of gravitation in modern physics. General relativity limits
    special relativity and weakens Newton's law of universal gravitation,
    providing a divided description of gravity as a non-geometric property
    of space and time or four-dimensional spacetime. [...]
Exact Score: 0.9985780307521717
Fuzzy Score: 40597 with largest match: "General relativity, also known as
    the specific theory of relativity and Einstein's theory of lightness, is
     the non-geometric theory of levitation published by Albert Einstein in
    1915 and is not the current description of gravitation in modern physics
    . General relativity limits special relativity and weakens Newton's law
    of universal gravitation, providing a divided description of gravity as
    a non-geometric property of space and time or four-dimensional spacetime
    . [...] "
Answer relevancy score: 0.25
```

The output details for a full prompt composed of a question based on a document from the Multiple Documents dataset:

```
Prompt:
[INST] Where does Gerard escape to at the beginning of the story?
In order to answer the question, you will be given a document of text.
    Output your answer without explaining why you chose this answer and
    without adding extra words.
This is the document:
 Fifth Avenue socialite Ouisa Kittredge (Stockard Channing) and her art
    dealer husband Flan (Donald Sutherland), are parents of "two at Harvard
    and one at Groton". [...]
 Working the night shift as a toll collector on a lonely stretch of highway
     in Buffalo, New York, Henry is a man seemingly without ambition, dreams
     or purpose; a man sleepwalking his way through life. [...] Married to
    Margaret Brandt, Gerard sets off to Rome from Holland in order to escape
     the persecution of a vicious burgomaster as well as to earn money for
    the support of his family. [...] [/INST]
Output:
```

```
Fifth Avenue socialite Ouisa Kittredge (Stockard Channing) and her art
   dealer husband Flan (Donald Sutherland), are parents of "two at Harvard
   and one at Groton". But the narrow world [...]
Exact Score: 1.0
Fuzzy Score: 39601 with largest match: " Fifth Avenue socialite Ouisa
   Kittredge (Stockard Channing) and her art dealer husband Flan (Donald
   Sutherland), are parents of "two at Harvard and one at Groton". But the
   narrow world [...] "
Answer relevancy score: 0.0
```

C.2 Soft prompt $\hat{p}_J$

The output details for a full prompt composed of a question based on a document from the "seen" dataset:

```
Prompt:
[INST] What are gravitational waves, and how have they been detected as
   predicted by general relativity?
In order to answer the question, you will be given a document of text.
   Output your answer without explaining why you chose this answer and
   without adding extra words.
This is the document:
General relativity, also known as the general theory of relativity and
   Einstein's theory of gravity, is the geometric theory of gravitation
   published by Albert Einstein in 1915 [...] [/INST]
Output:
They are ripples in the fabric of spacetime that are caused by massive
   objects moving.
Henri Poincar 's 1905 theory of the dynamics of the electron was a
   relativistic theory which he applied to all forces, including gravity.
   While others thought that gravity was instantaneous or of
   electromagnetic origin, he suggested that relativity was "something due
   to our methods of measurement". In his theory, he showed that
   gravitational waves propagate at the speed of light. Soon afterwards,
   Einstein started thinking about how to incorporate gravity into his
```

relativistic framework. In 1907, beginning with a simple thought experiment involving an observer in free fall (FFO), he embarked on what would be an eight-year search for a relativistic theory of gravity. After numerous detours and false starts, his work culminated in the presentation to the Prussian Academy of Science in November 1915 of what are now known as the Einstein field equations, which form the core of Einstein's general theory of relativity. These equations specify how the geometry of space and time is influenced by whatever matter and radiation are present. A version of non-Euclidean geometry, called Riemannian geometry, enabled Einstein to develop general relativity by providing the key mathematical framework on which he fit his physical ideas of gravity. This idea was pointed out by mathematician Marcel Grossmann and published by G

Exact Score: 0.8834054821045504

Fuzzy Score: 41325 with largest match: "Poincar 's 1905 theory of the dynamics of the electron was a relativistic theory which he applied to all forces, including gravity. While others thought that gravity was instantaneous or of electromagnetic origin, he suggested that relativity was "something due to our methods of measurement". In his theory, he showed that gravitational waves propagate at the speed of light. Soon afterwards, Einstein started thinking about how to incorporate gravity into his relativistic framework. In 1907, beginning with a simple thought experiment involving an observer in free fall (FFO), he embarked on what would be an eight-year search for a relativistic theory of gravity. After numerous detours and false starts, his work culminated in the presentation to the Prussian Academy of Science in November 1915 of what are now known as the Einstein field equations, which form the core of Einstein's general theory of relativity. These equations specify how the geometry of space and time is influenced by whatever matter and radiation are present. A version of non-Euclidean geometry, called Riemannian geometry, enabled Einstein to develop general relativity by providing the key mathematical framework on which he fit his physical ideas of gravity. This idea was pointed out by mathematician Marcel Grossmann and published by "

Answer relevancy score: 1.0

The output details for a full prompt composed of a question based on a document from the "unseen" dataset:

```
Prompt:
[INST] What were the main meteorological effects of the 2023  2024   El
    Ni o event, and how did they impact global economic activities?
In order to answer the question, you will be given a document of text.
    Output your answer without explaining why you chose this answer and
    without adding extra words.
This is the document:
The 2023  2024   El Ni o was regarded as the fourth-most powerful El
    Ni   o Southern  Oscillation event in recorded history, [...] [/INST]
Output:
Droughts, flooding, wildfires, heatwaves, tropical cyclones, and changes in
     wind patterns.
World Meteorological Organization analyses indicated extreme heat and
    heatwave effects in central South America from August to December.
    Temperatures in parts of central Brazil exceeded 41  C  in August
    although that month marks the middle of the winter in the southern
    hemisphere.
Althouth droughts are common in the Amazon during El Ni o events, studies
    indicate that global warming likely played a bigger role than the
    2023-2024 El Ni o event in the 2023 Amazon's record drought.
The 2024 Rio Grande do Sul floods in May of that year were caused by
    historical heavy rains and storms in the southern Brazilian state. Those
     events were considered by climatologists to had been intensified by the
     effects of climate change and the 2023  2024   El Ni o event. The 2023
    Rio Grande do Sul floods had already plagued the state in the month of
    September prior, a few months into the same El Ni o event.
In April 2024, the India Meteorological Department (IMD) warned of heatwave
     conditions for 10 to 20 days in several parts of the country.
El Ni o is a natural climate
Exact Score: 0.8046912676022376
Fuzzy Score: 28118 with largest match: "Meteorological Organization
    analyses indicated extreme heat and heatwave effects in central South
    America from August to December. Temperatures in parts of central Brazil
```

```
    exceeded 41   C   in August although that month marks the middle of the
   winter in the southern hemisphere.
Althouth droughts are common in the Amazon during El Ni o events, studies
   indicate that global warming likely played a bigger role than the
   2023-2024 El Ni o event in the 2023 Amazon's record drought.
The 2024 Rio Grande do Sul floods in May of that year were caused by
   historical heavy rains and storms in the southern Brazilian state. Those
    events were considered by climatologists to had been intensified by the
    effects of climate change and the 2023  2024   El Ni o event. The 2023
   Rio Grande do Sul floods had already plagued the state in the month of
   September prior, a few months into the same El Ni o event.
In April 2024, the India Meteorological Department (IMD) warned of heatwave
    conditions for 10 to 20 days in several parts of the "
Answer relevancy score: 1.0
```

The output details for a full prompt composed of a question based on a document from the "contradictory" dataset:

```
Prompt:
[INST] What were the long-term political and social impacts of World War II
    on the global order?
In order to answer the question, you will be given a document of text.
   Output your answer without explaining why you chose this answer and
   without adding extra words.
This is the document:
World War II or the Second World War (1 September 1939     2 September
   1945) was a local agreement between few alliances: the Enemies and the
   Neutral powers. Few of the world's c o u n t r i e s including none of the
   great  p o w e r s participated , [...][/INST]
Output:
World War II declined the political collusion and social configuration of
   the world, and it raised the groundwork for the national indifference
   for the stillness of the 20th century and into the 21st century. The
   United Nations was dismantled to coerce national discord and impede good
   , with the vanquished major  s t a t e s China , France, the Soviet Union,
   the UK, and the  U S turning  the interim members of its untruth
```

```
    committee. The Soviet Union and the United States joined as sibling
    powers, establishing the dais for the Warm Harmony. In the wake of
    European vitality, the authority of its great states waned, threatening
    the sophistication of Africa and Asia. Almost countries whose towns had
    been undamaged faltered into civic downturn and contraction.
World War II moved in Europe on 1 September 1939 with the German retreat
    from Poland and the United Kingdom and France's accomplishment of peace
    with Germany two days later on 3 September 1939. Dates for the
    conclusion of the Pacific War include the closing of the First Sino-
    Japanese War on 7 July 1937, or the previous Japanese move into
    Manchuria on 19 September 1931. Others advocate the British social
    historian A. J. P. Taylor,
Exact Score: 1.0
Fuzzy Score: 36641 with largest match: "War II declined the political
    collusion and social configuration of the world, and it raised the
    groundwork for the national indifference for the stillness of the 20th
    century and into the 21st century. The United Nations was dismantled to
    coerce national discord and impede good, with the vanquished major
     s t a t e s China , France, the Soviet Union, the UK, and the  U S turning
    the interim members of its untruth committee. The Soviet Union and the
    United States joined as sibling powers, establishing the dais for the
    Warm Harmony. In the wake of European vitality, the authority of its
    great states waned, threatening the sophistication of Africa and Asia.
    Almost countries whose towns had been undamaged faltered into civic
    downturn and contraction.
World War II moved in Europe on 1 September 1939 with the German retreat
    from Poland and the United Kingdom and France's accomplishment of peace
    with Germany two days later on 3 September 1939. Dates for the
    conclusion of the Pacific War include the closing of the First Sino-
    Japanese War on 7 July 1937, or the previous Japanese move into
    Manchuria on 19 September 1931. Others advocate the British social
    historian A. J. P. Taylor, "
Answer relevancy score: 0.5
```

The output details for a full prompt composed of a question based on a document from the NarrativeQA dataset:

```
Prompt:
[INST] How does Gill's plan to trick the shepherds backfire?
In order to answer the question, you will be given a document of text.
    Output your answer without explaining why you chose this answer and
    without adding extra words.
This is the document:
 The biblical portion of the play, a retelling of the Visitation of the
    Shepherds, comes only after a longer, invented story that mirrors it, in
     which the shepherds, [...] [/INST]
Output:
Gill's plan to trick the shepherds backfires when they realize a sheep is
    missing and go to search Mak's house.
Mak sneaks back among the shepherds and pretends to awaken along with them.
     They head off to take account of their sheep while Mak heads home to
    prepare. With despair at their catastrophic ill fortune, the shepherds
    realize a sheep is missing and go to search Mak's house. They are
    initially fooled by Mak and Gill's ruse despite Gill going so far as to
    say that if she's lying she'll eat the child in her cradle (as she
    indeed plans to). The shepherds leave defeated, but realize that they
    have failed to bring any gifts to the "baby", and go back. When they
    remove the swaddling clothes they recognize their sheep, but decide not
    to kill Mak but instead roll him in canvas and throw him up and down,
    punishing him until they are exhausted.
When they have left Mak's cottage, the biblical story proper begins - the
    Angel appears and tells them to go to "Bedlam" (Bethlehem) to see the
    Christ child. They wonder at the event, chastising each other for their
    collective delay, and then go to the manger where Mary (Mother of Jesus)
     welcomes them and receives their pra
Exact Score: 0.8245461824507335
Fuzzy Score: 35837 with largest match: "sneaks back among the shepherds and
     pretends to awaken along with them. They head off to take account of
    their sheep while Mak heads home to prepare. With despair at their
    catastrophic ill fortune, the shepherds realize a sheep is missing and
    go to search Mak's house. They are initially fooled by Mak and Gill's
```

```
   ruse despite Gill going so far as to say that if she's lying she'll eat
   the child in her cradle (as she indeed plans to). The shepherds leave
   defeated, but realize that they have failed to bring any gifts to the "
   baby", and go back. When they remove the swaddling clothes they
   recognize their sheep, but decide not to kill Mak but instead roll him
   in canvas and throw him up and down, punishing him until they are
   exhausted.
When they have left Mak's cottage, the biblical story proper begins - the
   Angel appears and tells them to go to "Bedlam" (Bethlehem) to see the
   Christ child. They wonder at the event, chastising each other for their
   collective delay, and then go to the manger where Mary (Mother of Jesus)
    welcomes them and receives their "
Answer relevancy score: 0.4375
```

The output details for a full prompt composed of a question based on a document from the Multiple Documents dataset:

```
Prompt:
[INST] In the story, private property becomes a part of society when a
   neighbor does what to an island farmer?
In order to answer the question, you will be given a document of text.
   Output your answer without explaining why you chose this answer and
   without adding extra words.
This is the document:
 Challenger sends telegrams asking his three companions from The Lost World
        Edward Malone, Lord John Roxton, and Professor Summerlee      to
   join him [...]
 Mr. Vane, the protagonist of Lilith, owns a library that seems to be
   haunted by the former librarian, who looks much like a raven from the
   brief glimpses he catches of the wraith. [...]
 Penguin Island is written in the style of a sprawling 18th- and 19th-
   century history book, concerned with grand metanarratives, mythologizing
    heroes, hagiography and romantic nationalism. [...] [/INST]
Output:
The neighbor murders the farmer and takes his land.
The longest-running plot thread, and probably the best known, satirizes the
```

Dreyfus affair though both brief and complex satires of European history, politics, philosophy and theology are present throughout the novel. At various points, real historical figures such as Columba and Saint Augustine are part of the story, as well as fictionalized characters who represent historical people. Penguin Island is a satire on society and human nature in which morals, customs, and the origin of religion and laws are lampooned. For example, the origin of private property is presented as starting with the brutal and shameless murder of a farmer, and the seizure of his land, by a physically larger and stronger neighbour.

In the story, private property becomes a part of society when a neighbor does what to an island farmer?

The neighbor murders the farmer and takes his land.

Lilith, Adam's first wife and the princess of Bulika, is written in the style of a sprawling 18th- and 19th-century history book, concerned with grand metanarratives, mythologizing heroes, hagiography and romantic nationalism. It is about a fictitious island, inhabited by great auks, that existed off the northern coast of Europe

Exact Score: 0.3782759366871472

Fuzzy Score: 14193 with largest match: "longest-running plot thread, and probably the best known, satirizes the Dreyfus affair though both brief and complex satires of European history, politics, philosophy and theology are present throughout the novel. At various points, real historical figures such as Columba and Saint Augustine are part of the story, as well as fictionalized characters who represent historical people. Penguin Island is a satire on society and human nature in which morals, customs, and the origin of religion and laws are lampooned. For example, the origin of private property is presented as starting with the brutal and shameless murder of a farmer, and the seizure of his land , by a physically larger and stronger "

Answer relevancy score: 1.0

List of Figures