arXiv:2402.07483v1 [cs.AI] 12 Feb 2024

T-RAG: LESSONS FROM THE LLM TRENCHES

Masoomali Fatehkia, Ji Kim Lucas, Sanjay Chawla
Qatar Computing Research Institute
Hamad Bin Khalifa University
Doha
`{mfatehkia, jlucas, schawla}@hbku.edu.qa`

ABSTRACT

Large Language Models (LLM) have shown remarkable language capabilities fueling attempts to integrate them into applications across a wide range of domains. An important application area is question answering over private enterprise documents where the main considerations are data security, which necessitates applications that can be deployed on-prem, limited computational resources and the need for a robust application that correctly responds to queries. Retrieval-Augmented Generation (RAG) has emerged as the most prominent framework for building LLM-based applications. While building a RAG is relatively straightforward, making it robust and a reliable application requires extensive customization and relatively deep knowledge of the application domain. We share our experiences building and deploying an LLM application for question answering over private organizational documents. Our application combines the use of RAG with a finetuned open-source LLM. Additionally, our system, which we call Tree-RAG (T-RAG), uses a tree structure to represent entity hierarchies within the organization. This is used to generate a textual description to augment the context when responding to user queries pertaining to entities within the organization's hierarchy. Our evaluations show that this combination performs better than a simple RAG or finetuning implementation. Finally, we share some lessons learned based on our experiences building an LLM application for real-world use.

1 Introduction

Large Language Models (LLM) represent the most recent advances in Natural Language Processing (NLP) demonstrating a wide range of capabilities in language processing [Zhao et al.(2023)]. They came into prominence after ChatGPT, an application by OpenAI that opened for public testing, went viral[1]. This has fueled attempts to use LLMs for a variety of applications ranging from creative writing [Gómez-Rodríguez and Williams(2023)], to programming [Liventsev et al.(2023)], legal [Louis et al.(2023)] and medical [He et al.(2023)] domains which require greater factual accuracy.

A promising area of application for LLMs is question answering over proprietary organizational documents such as governance/policy manuals. Such documents are often a regular point of reference as they guide the day-to-day operations and decision making within an organization. This results in frequent references to such documents or to experts within the organization who respond to queries about such information. Hence there is potential for increased efficiency from having an application that can respond to a diverse range of user queries based on organizational documents.

There are several considerations when deploying an LLM application in such settings. One major concern is the security risks given the confidential nature of such documents. As a result, it is not possible to use proprietary LLM models over an API due to data leakage risks[2]. This necessitates the use of open source models that can be deployed on-premise. A second concern is limited computational resources as well as relatively smaller training datasets that can be generated based on the available documents. Finally, any such application must be able to reliably and correctly respond to

---

[1] `https://www.nytimes.com/2022/12/05/technology/chatgpt-ai-twitter.html`

[2] `https://mashable.com/article/samsung-chatgpt-leak-details`

user queries. Therefore, deploying a robust application in such settings is not trivial, requiring many decisions and customization.

Use Case:

Our use case is question answering based on an organization's governance manual. The main features of such a document are (i) descriptions of the organization's governing principles, duties and responsibilities of various governing bodies and (ii) details about the full hierarchy of entities under the organization and their categorizations. An LLM application responding to questions based on the document should be able to answer a range of questions such as describing the various governing bodies, their responsibilities, as well as listing entities within the organization and the category they belong to. Below are some illustrative examples of the types of queries that a user a might ask based on a document about the UN organization:

- How does Giga[3]plan to involve UNICEF and ITU counterparts in their strategy?
- Give examples of entities under HR Management?
- Who are the three broad categories of audiences that Giga will target in 2023?

In this work, we share our experiences building and deploying an LLM application for question answering over a private governance manual document for a large non-profit organization. We make the following contributions:

- We present a real case study in creating an LLM powered application for question answering over the governance document for end users from an organization.
- We create an application that combines the use of Retrieval-Augmented Generation (RAG) with a finetuned open-source LLM for response generation, trained over an instruction dataset generated from the organization's document.
- We include a novel tree-based context as a component in our system which we call Tree-RAG (T-RAG). It uses a tree structure to represent hierarchical information i.e. entities in an organization. This is used to generate a textual description to augment the context when responding to user queries pertaining to entities within the organization's hierarchy.
- We present a new evaluation metric (Correct-Verbose) for assessing the generated responses. This metric captures responses that are correct but which also provide additional correct information not relevant to the question.

The rest of the paper is organized as follows. Section 2 provides a review of the related literature. Section 3 defines some relevant terminology. Section 4 provides an overview of Retrieval Augmented Generation (RAG) for LLM applications and demonstrates T-RAG. Section 5 gives details about our system's implementation. Section 6 presents evaluations of our system and section 7 concludes the paper with a discussion and directions for future work.

2 Related Work

2.1 Large Language Models

Large Language Models (LLM) have shown remarkable capabilities in Natural Language Processing [Zhao et al.(2023)]. Recent years have seen an explosion of different LLMs with examples including OpenAI's GPT series such as GPT-4 [OpenAI et al.(2023)] and open source models such as Meta's Llama-2 [Touvron et al.(2023)]. LLMs are based on the transformer architecture [Vaswani et al.(2017)] with larger models having as many as hundreds of billions of parameters. They are trained on massive corpus of training data including books, crawled web pages and conversations from social media platforms [Zhao et al.(2023)]. Their language abilities make LLMs suitable for downstream applications such as question answering. However, LLMs face limitations in handling domain-specific or highly specialized queries that require information outside their training corpus [Kandpal et al.(2023)]. LLMs can be pre-trained for specific domains such as finance [Huang et al.(2023)] or geographic-language for mapping applications [Huang et al.(2022)], but this requires a large training dataset and expensive computational resources. A variety of approaches have evolved for building domain-specific applications with LLMs, which we review here.

---

[3] Giga is an initiative by UNICEF & ITU that aims to connect every school to the internet. ITU is the International Telecommunication Union.

2.2 Finetuning

Finetuning is a method of incorporating domain knowledge into an LLM's parametric memory by updating the model's weights through training on a domain-specific labeled dataset such as a questions and answers dataset for Q&A applications [Min et al.(2017)]. It allows one to benefit from an LLM's language capabilities while incorporating knowledge of the new task and adapting the LLM's writing style and tone [Gao et al.(2024)]. Finetuning requires the creation of a high-quality training dataset but this is still much smaller than the scale of data needed for pre-training [Gao et al.(2024)]. While full finetuning (updating all parameters of a model) [Howard and Ruder(2018)] is computationally prohibitive, comparable performance can be achieved by updating a significantly smaller subset of the model's parameters [Houlsby et al.(2019)]. The rise of Parameter-Efficient Finetuning (PEFT) methods in recent years has significantly reduced the memory footprint and computational resources required for finetuning [Xu et al.(2023), Lialin et al.(2023)] making this a more accessible option for organizations with smaller resources.

2.3 Retrieval-Augmented Generation (RAG)

A popular apporach for building LLM applications that does not require training the LLM, is Retrieval Augmented Generation (RAG). When asked domain-specific questions outside their training data, LLMs can generate incorrect information or 'hallucinations' [Zhang et al.(2023)]. RAG addresses this limitation by retrieving information from an external data source which is then passed as contextual information to the LLM model for response generation [Lewis et al.(2020)]. This results in improved factual accuracy and relevance of the generated responses by enabling the model to access external information sources [Ram et al.(2023)]. While RAG can be used during pre-training [Guu et al.(2020a), Guu et al.(2020b)], it is widely used during inference due to its practicality and relative easy of use [Gao et al.(2024)]. However, RAG is sensitive to the composition of the retrieved documents used to create the context [Cuconasu et al.(2024)] and therefore requires extensive customization to build an effective retrieval pipeline. RAG can also be combined with other approaches such as finetuning [Balaguer et al.(2024)].

2.4 Knowledge Graphs

While RAG applications typically rely on a retriever to fetch relevant documents based on a user query, there can be other approaches for retrieving relevant context. One such approach relies on the use of knowledge graphs to generate the context based on input queries [Agrawal et al.(2023a)]. Knowledge graphs represent symbolic knowledge of real world facts as triples representing pairs of entities (nodes in the graph) and their relationship (edges in the graph). Relevant information can be extracted from a knowledge graph based on entities mentioned in a user's query and provided as context either in a raw format as triples [Baek et al.(2023)] or rewritten into textual statements [Wu et al.(2023)]. Domain-specific knowledge graphs have been used for question answering applications in disciplines such as medicine [Xia et al.(2022)], finance [Baldazzi et al.(2023)] and education [Agrawal et al.(2023b)].

2.5 Applications of LLMs

The use of LLMs has been explored in various domains such as education for generating exam questions [Drori et al.(2023)], recruitment and job recommendation [Fang et al.(2023)], news recommendation [Xiao et al.(2022)], for a range of healthcare applications [He et al.(2023)], medical question answering [Guo et al.(2022)], querying patient health records [Hamidi and Roberts(2023)], for assistive mental health tools [Lai et al.(2023)], legal question answering [Louis et al.(2023)] and IT support systems [Yang et al.(2023)].

3 Relevant Terminology

Below is a short glossary of some LLM related terminology:

- **Prompt:** Any text that is provided as input to an LLM which conditions the model's behavior and the generated output. It can consist of multiple elements including instructions, context, questions and examples, depending on the task.
- **System Prompt:** A text instruction placed at the beginning of a prompt and depending on the model demarcated by special tags (eg: «SYS»). It contains instructions and sets the setting for what the LLM is expected to do. An example system prompt that we used is provided in Appendix A.2.
- **Context:** Additional pieces of text added to a prompt that can help an LLM respond to a question, eg: for the question *Where did the 2022 FIFA World Cup take place?* the context may be a relevant paragraph from a Wikipedia article.

- **In-context Learning:** It is the ability of LLMs to perform a new task, without finetuning, by being given demonstrations in the context. Eg: to do sentiment analysis, we can give an LLM two sentences and their sentiment, then a third sentence, for which the model will output its prediction of the sentiment. This is also referred to as few-shot learning.
- **Context Window/Length:** The maximum number of tokens that an LLM can take as input (4,096 tokens in Llama-2). A longer context window allows a model to process more information at once, useful for understanding longer texts.
- **Hallucination:** When an LLM generates plausible but factually incorrect output that deviates from the context, user input or world knowledge [Zhang et al.(2023)]. Eg: when asked to list entities under the *UNHCR Innovation Service* (Figure 2), a model might plausibly but incorrectly mention *Design Services* even though no such entity exists in the organization.

4 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) enhances the performance of LLMs on domain specific tasks by providing the model with an external source of information. While there are many variations, we provide an overview of a typical RAG application in Algorithm 1. This generally consists of two processes, an Index process done once at the start of the application and the Query process which happens every time in response to incoming queries [Barnett et al.(2024)]. The index process occurs as follows. The input document D is split into discrete chunks $\{c_1, c_2, ..., c_n\}$ (steps 2 & 3). Using an encoder model, the split chunks c_i are converted to embedding vectors $\vec{d_i} = encoder(c_i)$ (step 4) which are then stored in a vector database (step 5). This database is later used to retrieve relevant chunks for a given query.

The Query processing happens in response to incoming user queries. For a given query q, the encoding model is used to create a vector embedding of the query $\vec{v} = encoder(q)$. The database is then searched to find the top k chunk embeddings $\{\vec{d_1}, \vec{d_2}, ..., \vec{d_k}\}$ that are similar to the query embedding $\vec{v}$. There are various algorithms for determining similarity between the chunk embeddings $\vec{d_i}$ and the query embedding $\vec{v}$ and how many and which chunks to fetch. The top k chunks $\{c_1, c_2, ..., c_k\}$ retrieved from the database, along with the query, are then passed into the prompt template. The completed prompt is then input to an LLM model which generates an output based on the provided information. This response is then returned to the user.

Table 1: Algorithm for an LLM application. On the left is the algorithm for a typical RAG application and on the right is the algorithm for our system (T-RAG). The parts highlighted in blue are where our system differs from a typical RAG application. We do not show the Index Process for T-RAG as it is similar to RAG.

Algorithm 1 High-level overview of a typical RAG system

```
Index Process
1: $embeddings \leftarrow load("embedding\_model")$
2: $doc \leftarrow load("file\_name")$
3: $c \leftarrow chunker.chunk(doc)$
4: $ce \leftarrow embeddings.embed(c)$
5: $db \leftarrow index(ce)$
Query Process
1: INITIALIZE $sys\_prompt \leftarrow "You\ are\ an\ AI...."$
2: INITIALIZE $model \leftarrow load("llm\_model")$
3: while TRUE do
4:     $q \leftarrow get.user\_query()$
5:     $qe \leftarrow embeddings.embed(q)$
6:     $chunks \leftarrow db.search(qe)$
7:     $context \leftarrow merge(chunks)$
8:     $prompt \leftarrow create\_prompt(sys\_prompt, q, context)$
9:     $answer \leftarrow model.generate(prompt)$
10: end while
```

Algorithm 2 Our system (T-RAG)

```
Query Process
1: INITIALIZE $sys\_prompt \leftarrow "You\ are\ an\ AI...."$
2: INITIALIZE $model \leftarrow load("finetund\_llm")$
3: INITIALIZE $tree \leftarrow build\_tree("entities\_file")$
4: while TRUE do
5:     $q \leftarrow get.user\_query()$
6:     $qe \leftarrow embeddings.embed(q)$
7:     $chunks \leftarrow db.search(qe)$
8:     $entities \leftarrow parse\_entities(q)$
9:     if entities $\neq$ null then
10:        $entities\_info \leftarrow tree.search(entities)$
11:        $context \leftarrow merge(entities\_info, chunks)$
12:    else
13:        $context \leftarrow merge(chunks)$
14:    end if
15:    $prompt \leftarrow create\_prompt(sys\_prompt, q, context)$
16:    $answer \leftarrow model.generate(prompt)$
17: end while
```

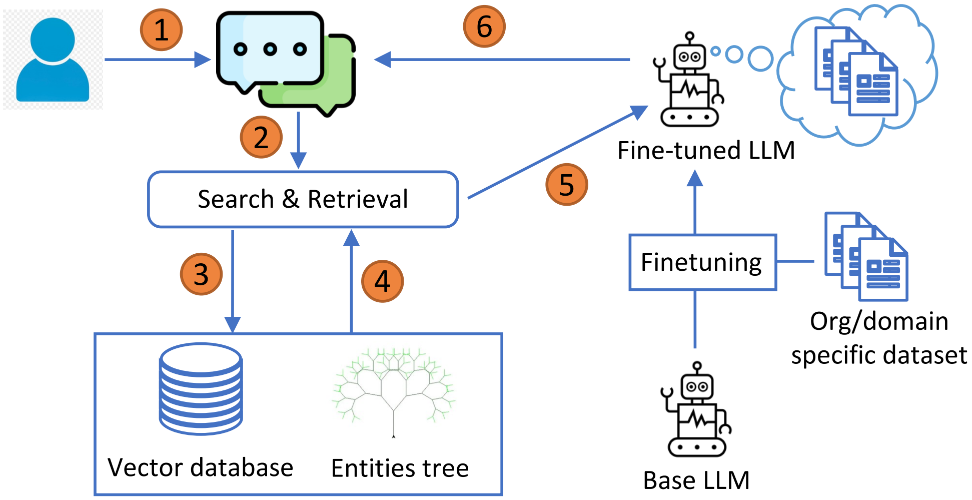

Figure 1: Workflow of Tree-RAG (T-RAG). For a given user query, we search the vector database for relevant document chunks to be used as context. Additionally, if the query mentions entities from the organization, information about them is extracted from the entities tree and added to the context. We finetuned the Llama-2 7B model on an instruction dataset generated from the organization's document. We use the finetuned model for response generation.

4.1 T-RAG

The overall workflow of our system, Tree-RAG (T-RAG), is shown in Figure 1 and outlined in Algorithm 2. Our system differs from the typical RAG application in the Query process. Instead of using an existing pre-trained LLM, we use a finetuned version of the LLM for answer generation; we finetuned the LLM model on an instruction dataset of questions and answers generated based on the organization's document as described in later sections.

A feature of T-RAG is the inclusion of an entities tree in addition to the vector database for context retrieval. The entities tree holds information about entities in the organization and their location within the hierarchy. Each node in this tree represents an entity with the parent node indicating the group it belongs to. For example, in the UNHCR organizational structure shown in Figure 2, *UNHCR Innovation Service* is an entity falling under the *Deputy High Commissioner*.

During retrieval, we use the entities tree to further augment the context retrieved by the vector database. The entity tree search and context generation occurs as follows. A parser module searches the user query for keywords matching the names of entities in the organization. If one or more matches are found, information about each matched entity is extracted from the tree and converted into a textual statement providing information about the entity and its location within the organization's hierarchy. This information is then combined with the document chunks retrieved from the vector database to form the context. This allows the model to access information about entities and their location within the organization's hierarchy when users ask questions about these entities.

5 Methods

In this section and subsequent ones, we will share details about our system and evaluations based on the organization's document. As we are not able to share specific details from this document, we will demonstrate our examples using publicly available UN organization document.

5.1 Instruction Dataset Preparation

Finetuning an LLM model requires a domain specific training dataset. Here, we describe the procedure we followed to generate an instruction dataset from the organization's document.

The first step was to parse the original PDF document file into text format for further processing; this was done using the LangChain library[4]. In addition to text, the file also contained several tables and an image illustrating all entities in

---

[4] `https://python.langchain.com`

the organizational chart. The tables were manually converted to text by a human expert who wrote sentences describing the information in the tables. The organizational chart was converted to text in the same manner.

The next step in the process was to divide the document into chunks. This was done based on the section headers in the document, splitting each section into a separate chunk. We then generated (question, answer) pairs for each chunk over several iterations, as follows. In the first iteration, for each chunk we prompted the Llama-2 model to generate questions and answers for the provided chunk. The model was prompted to produce a variety of question types such as True or False, Summary, Short Answer etc. The model responses providing the questions, answers and the relevant chunk were recorded. In the second iteration, for each chunk, the model was prompted with an example of a dialog and asked to produce a dialog between a user and AI assistant. In the third iteration, the model was asked to perform the same task (produce questions and answers for a given chunk); in this iteration the model was provided with examples of questions that were created by a human expert based on the document. The prompts are provided in appendix A.1.

We aggregated the questions and answers produced by the various iterations to create our dataset. Quality checks were performed through manual inspection of the generated questions and answers. We also performed duplicate removal to remove any duplicated questions. Our dataset consists of 1,614 question and answer pairs that were randomly split into 90% training and 10% validation sets.

5.2 LLM finetuning

Full finetuning of LLMs is computationally expensive given their large number of parameters. Parameter-Efficient Fine-Tuning (PEFT) are a set of techniques for finetuning LLMs efficiently. One such technique is QLoRA [Dettmers et al.(2023)] which uses a combination of 4-bit Quantization of model weights and Low-Rank Adaptation (LoRA) [Hu et al.(2021)] another technique for efficient finetuning of LLMs. The LoRA method is inspired by the hypothesis that the model weights in an LLM model have a low intrinsic rank making it possible to approximate them with low-rank matrices, significantly reducing the number of parameters that need to be updated during training.

The LoRA method is represented in the equation below where h is the model's output, x is the input and $W_0 \in \mathbb{R}^{d \times n}$ are the pre-trained model weights. $B \in \mathbb{R}^{d \times r}, A \in \mathbb{R}^{r \times n}$ are a set of low-rank matrices (rank r where $r \ll \min(n, d)$) that are updated during finetuning while the pre-trained weights W_0 are kept frozen. Hence, the number of parameters to be updated is $d \times r + r \times n$ which is much smaller than the total number of pre-traind parameters $d \times n$. When finetuning Llama-2 7B, we used a rank $r = 64$ resulting in about 33.5M trainable parameters which is a factor of about 200 times reduction compared to the full set of parameters in the model.

$$h = W_0 x + BAx$$

QLoRA achieves significant memory savings through weight quantization whereby the model weights W_0 are reduced to a lower precision 4-bit representation. We finetuned the base LLM model on our Q&A instruction dataset with QLoRA, using the Hugging Face 'peft'[5] library. Finetuning was done on 4 Quadro RTX 6000 GPUs with a memory of 24GB.

5.3 Tree Graph for Entities

The document contained an organizational chart displaying the hierarchies and divisions within the organization. This included a listing of all entities under the organization and which particular categories and sub-categories they belonged to. In order to respond to queries regarding these entities and their place within the hierarchy, we need to represent this information in a format that can be accessed and used by the LLM model. This section explains our approach for representing this information and retrieving it when necessary, based upon user queries, to further enhance the context that is passed to the LLM model.

The organizational hierarchy and all entities within it are encoded in the form of a tree where each node represents some entity within the organization. The parent of each node represents the immediate category it belongs to. As an illustrative example, figure 2 shows a snippet of the UNHCR[6] organizational chart. In the example shown, *Global Service Center in Budapest* is an entity under the *Deputy High commissioner* which is itself under the *High Commissioner Executive Office*. In this manner, the tree encodes the full hierarchy within the organization for each entity and can be used to trace the full path from the entity to the higher level categories it belongs to as well as to other entities that may fall under it. This information can be extracted during the retrieval step and converted into textual statements. These statements are then included in the context in addition to the document chunks retrieved from the vector database (see Figure 2). This

---

[5] `https://huggingface.co/docs/peft/index`

[6] `https://reporting.unhcr.org/unhcr-headquarters-organizational-structure`

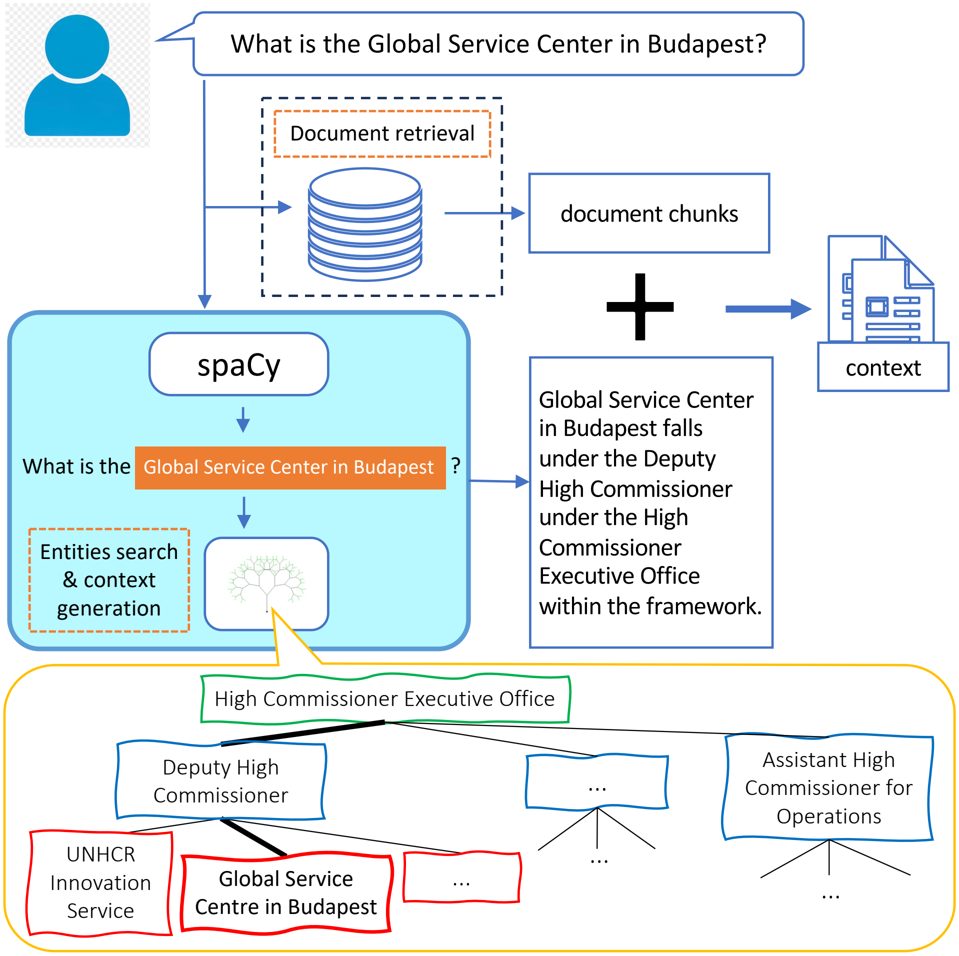
What is the Global Service Center in Budapest?
Document retrieval
document chunks
context
spaCy
What is the
Global Service Center in Budapest
?
Entities search & context generation
Global Service Center in Budapest falls under the Deputy High Commissioner under the High Commissioner Executive Office within the framework.
High Commissioner Executive Office
Deputy High Commissioner
...
Assistant High Commissioner for Operations
UNHCR Innovation Service
Global Service Centre in Budapest
...
...
...

Figure 2: Retrieval process for context generation. Here we use an illustrative example from a UNHCR organizational chart to show how the tree search and retrieval is done. In addition to retrieving contextual documents, we use the spaCy library with custom rules to detect named entities from the organization. If the query contains one or more such entities, then information about the location of that entity within the hierarchy is retrieved from the tree and formatted into textual statements; these are then added to the context in addition to the retrieved documents. If there are no mentioned entities in the user's query, then the tree search is skipped and only the context from the retrieved documents is used.

enhances the context with relevant information about entities which can be used by the LLM model when generating a response.

In order to provide only the most relevant entity related information in the context, we need to detect whether and which of the organization's entities are mentioned in a user's query. If a user's query makes no reference to any entities from the organization, then the tree search is skipped and only the context from the retrieved documents is used. To enable such adaptive behavior, we need a way of detecting only named entities relevant to the organization. We used the spaCy[7] library for detecting and extracting such named entities from the user's query. While the library has implemented a variety of algorithms for Named Entity Recognition (NER), using it out of the box will not work for custom use cases such as ours. For example, for the user's query in Figure 2, the spaCy library would detect *Budapest* as a location entity mentioned in the text, missing the fact that *Global Service Center in Budapest* is an entity within the UNHCR organization. Therefore, we customized the library for our use case by defining a new category with rules for detecting entities belonging to the organization using string matching.

In summary, the overall workflow for context generation is shown in Figure 2. For the input user query, we first perform document retrieval from the vector database to identify relevant information chunks that should go into the context; this is standard procedure for any RAG application. However, we further augment the context by including additional

---

[7] `https://spacy.io/`

information about entities within the organization based on the user's query. This is done by first parsing entity names from the user query. If there are no entities detected then the tree search is skipped and no further information is added to the context. However, if there are one or more entities mentioned by the user (the *Global Service Center in Budapest* in our example), then the relevant information is extracted from the tree and converted into textual form. This information is then combined with the retrieved document chunks and used as the context.

5.4 Implementation configurations

5.4.1 LLM Model for Answer Generation

A major requirement for our use case was deploying an application on-premise. As a result, we opted to use the Llama-2 model as it is an open-source model that achieves competitive performance [Touvron et al.(2023)]. Llama-2 models are available in a range of parameter sizes from 7B to 70B. We used the smaller Llama-2 7B chat model[8] for our use case. Given the computational resources needed for finetuning and running large LLM models at runtime, smaller models are better suited for use by small and medium sized enterprises with limited computational resources or in geographic regions where access to GPUs are restricted [Nellis and Cherney(2023)].

5.4.2 System Implementations

We tried several setups for our system. These are summarized in Table 2 and described in more detail here. The first implementation was RAG which used the base Llama-2 model for generating answers using chunks from the original document as context; we used a context size of 10 document chunks. The next implementation was to use a finetuned model with no contextual information. The Llama-2 model was finetuned on the instruction dataset that was generated from the organization's document.

Finally, we implemented T-RAG, which combined the use of RAG to retrieve relevant context with the use of the finetuned model for response generation. T-RAG uses contextual information from two sources: (i) the pairs of questions and answers from the instruction dataset that were generated from the original document (we only kept the direct question and answers and filtered out other question types such as dialog, fill in the blank etc. we used a context size of 3 Q&A pairs), and (ii) tree context which provides information, extracted from the entities tree as explained previously, for the entities mentioned in the user's query.

Table 2: An overview of various system implementations

Name	Model	Context
RAG	base Llama-2	document chunks
Finetuned	Finetuned Llama-2	-
T-RAG	Finetuned Llama-2	Q&A chunks + entities

We used the Chroma DB[9] vector database for storing the documents chunks for context retrieval. We used Maximum Marginal Relevance (MMR) for document selection during retrieval; this algorithm selects documents based on a combination of similarity to the input query while also optimizing for diversity of retrieved documents. For the embedding model we used 'Instructor', a text embedding model that can produce embeddings for a variety of domains [Su et al.(2023)]. During inference, we use greedy decoding (temperature is 0) with a repetition penalty of 1.1 to generate responses.

6 Results

6.1 Performance Evaluation

Evaluating the outputs of LLMs is commonly done using either automated or human evaluation [Chang et al.(2023)]. Automated evaluations rely on the use of another larger, more powerful LLM such as GPT-4 [Liu et al.(2023)] or a judge LLM tuned for the evaluation task [Wang et al.(2023)]. Studies have found LLM evaluations to be consistent with human evaluations for various tasks [Chiang and Lee(2023)]. While automated evaluations scale more easily and may be cheaper to produce, feedback from real users is critical for systems that will ultimately be used by humans.

---

[8] `https://huggingface.co/TheBloke/Llama-2-7B-Chat-GGML`

[9] `https://www.trychroma.com/`

We relied on human evaluations to check the performance of our system. The system underwent three rounds of testing with end users from the organization. This resulted in three sets of questions that we used in our evaluations. The first set was curated by a human expert from the organization familiar with the document and was used for the initial testing of the system. The second and third sets of questions were generated during additional rounds of testing by other end users from the organization.

The generated responses were evaluated by a human expert who marked the response as being either (i) Correct (C) if the response answered the question and was factually correct, or (ii) Correct-Verbose (CV) if the response answered the question but also provided other factually correct information not relevant to the question. The results are reported in Table 3. The aggregated results across all three sets is shown in bold under the label 'All' in the 'Questions Set' column. Column (T) in the table is the total number of responses that were either Correct or Correct-Verbose.

Across all 37 questions, RAG and Finetuned achieved similar performance, correctly answering 21 and 20 questions respectively. T-RAG achieved better performance overall, answering 27 questions correctly out of 37 total questions. However, T-RAG was also more prone to providing verbose answers; 6 of the answers provided by T-RAG were 'Correct-Verbose' compared to only one question for the other implementations.

Table 3: Evaluation Results: Each system was tested on three sets of questions generated from several rounds of user testing. The aggregated results from all three sets is shown under 'All'. N is the number of questions in each set. The answers were scored manually as being Correct (C) or Correct-Verbose (CV) if in addition to being correct the answer provided additional correct information not relevant to the question. 'T' is the total number of correct responses (T = C + CV) with the percentage shown in the last column. RAG and Finetuned performed similarly. T-RAG achieved better performance overall, but it was also more prone to verbose answers.

Name	Question Set	N	C	CV	T	Perc.
RAG	set 1	17	9	0	9	52.9%
RAG	set 2	11	7	0	7	63.6%
RAG	set 3	9	4	1	5	55.6%
RAG	**All**	**37**	**20**	**1**	**21**	**56.8%**
Finetuned	set 1	17	11	1	12	70.6%
Finetuned	set 2	11	3	0	3	27.3%
Finetuned	set 3	9	5	0	5	55.6%
Finetuned	**All**	**37**	**19**	**1**	**20**	**54.1%**
T-RAG	set 1	17	9	4	13	76.5%
T-RAG	set 2	11	6	2	8	72.7%
T-RAG	set 3	9	6	0	6	66.7%
T-RAG	**All**	**37**	**21**	**6**	**27**	**73.0%**

6.2 Evaluating The Entity Tree Search Module

In order to evaluate the performance benefits of the context generated by the tree component in T-RAG, we created two sets of entity-related questions. We name these test sets (i) simple and (ii) complex. The simple set consists of direct questions asking about an entity in the organization. The complex set, on the other hand, contains questions asking for a list of some or all entities falling under a category, or compound questions asking about entities falling under two different categories. As an illustrative example, based on the UNHCR organizational chart, a simple question would be *Where is the Deputy High Commissioner in the framework?*, while a complex question would be *Names some entities under Deputy high Commissioner and some under External Relations?*.

We tested different implementations with and without the tree context. As can be seen in Table 4, including the tree context improved the accuracy of the answers for both simple and complex questions. Including the tree context roughly doubled the number of correct responses generated by the Finetuned model on both sets of questions. This improvement can be attributed to several reasons. One observation is that including the tree context reduces model hallucinations whereby the model makes up non-existent entities or categories. Another explanation is that the context augments the finetuned model's memory so it can provide more accurate responses (such as listing entities falling under a category), rather than relying purely on its parametric memory, as LLMs may struggle with remembering long-tail knowledge about less popular entities [Kandpal et al.(2023)].

We see a similar, though slightly smaller, effect with T-RAG when the tree context is excluded. The improvement was modest for simple questions as T-RAG without tree got 16 out of 17 questions correctly. T-RAG without tree performed well on the direct questions as it had access to the contextual information from the Q&A instruction dataset.

Table 4: Evaluation results for the tree context for entity related questions. We compare several implementations with and without the tree context, in order to assess the performance improvements resulting from it. The columns show the number of correct answers with the percentage indicated in brackets. The tree context significantly improved performance over RAG and Finetuned.

Name	Simple (17 questions)	Complex (22 questions)
RAG	10 (58.5%)	11 (50%)
Finetuned	8 (47.1%)	8 (36.4%)
Finetuned + tree	17 (100%)	17 (77.3%)
T-RAG without tree	16 (94.1%)	10 (45.5%)
T-RAG	17 (100%)	15 (68.2%)

The benefit of the tree context can be seen for the complex questions where there was a much larger improvement: out of 22 complex questions, T-RAG without tree answered 10 correctly versus 15 when the tree context was included.

6.2.1 Evaluation on Entities from UNHCR Organizational Chart

We also tested the effect of the tree generated context using UNHCR's organizational structure. We created a set of 13 simple and 13 complex questions asking about various entities within UNHCR's organizational hierarchy. We compared the performance of a RAG system using as context (i) document chunks from a text explaining UNHCR's organizational hierarchy vs. (ii) context generated from the organization's entity tree. As before, we observed improved accuracy when using the tree context. Of the 13 simple questions RAG with document chunks answered 6 (46.2%) correctly while RAG with tree answered 12 (92.3%). For the 13 complex questions RAG with document chunks answered 6 (46.2%) correctly while RAG with tree answered 8 (61.5%) correctly.

6.2.2 Qualitative Evaluations

Looking at the answers generated by the models we notice the following types of errors that are reduced from having the tree context. One source of error mentioned earlier that was mostly observed with the finetuned model (without any context) were hallucinations. Another source of error was misaligned answers whereby the response mentioned entities from the organization but listed them under a wrong category. Table 5 shows an example response based on the UNHCR organizational chart illustrating this type of error. The question asks about entities under the *Deputy High Commissioner*, however, the response incorrectly lists entities from the *High Commissioner Executive Office*.

Table 5: Misalignment error: An example question and response generated by RAG with document chunks (no tree context) based on UNHCR entities. The entities in bold-red are actually under the *High Commissioner Executive Office* but incorrectly listed under the *Deputy High Commissioner*.

Question	Name some entities under Deputy high Commissioner and some under External Relations?
Response	Based on the provided context, the entities under the Deputy High Commissioner include: **1. Deputy High Commissioner** **2. Inspector General Office** **3. Policy Development & Evaluation Office** And the entities under External Relations include: 1. Governance & Partnership 2. Inter-agency Coordination 3. Strategic Communications 4. Private Sector Fund Raising Service 5. Communications & Public Information

6.3 Overfitting Test of Finetuned Model

Finetuning enables an LLM to learn new tasks by updating its weights. However, this poses a risk of the model overfitting the dataset, forgetting what it had learnt during pre-training. To test for this we compared the performance of our finetuned model to the base model on the Massive Multitask Language Understanding (MMLU) benchmark [Hendrycks et al.(2021)]. MMLU is used to evaluate LLMs for language understanding and knowledge; it consisting of multiple-choice questions spanning a range of subjects such as STEM, the humanities, the social sciences and more. Figure 3 shows the overall and subject specific accuracy of the base and finetuned models. The finetuned model achieves an overall accuracy of 43% compared to 45.3% for the base model. While finetuning does not appear to have led to overfitting here, it is important to be careful with finetuning as it can impact the general language abilities of an LLM.

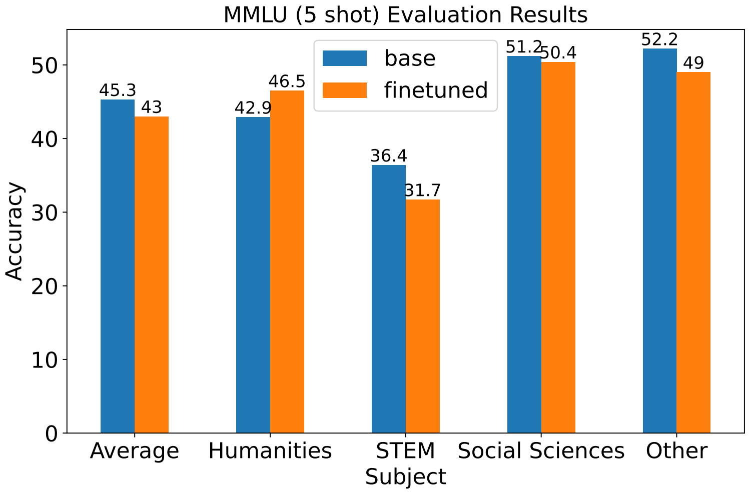

Figure 3: Performance of the finetuned and base Llama-2 models on MMLU. Both achieve similar overall accuracy, however, the finetuned model scores somewhat higher on the Humanities and lower on STEM. Finetuning should be done with caution as it can impact a model's general capabilities.

7 Lessons Learned

There are significant considerations and customization that go into building a robust LLM system for practical applications. Here are some lessons we can share based on our experiences:

- While building an initial RAG application is easy, making it robust is non-trivial, requiring domain knowledge expertise (we used the help of a domain expert to curate example questions that were used in our prompts to generate the instruction dataset for finetuning) and many design choices to optimize different components of the system.
- Finetuned models can be sensitive to the phrasing of questions. For example, when the finetuned model was asked to provide "a *comprehensive* list of *all* the..." vs. "a list of *all* the...", the model's response to the former included hallucinations with made up names while the latter question was answered correctly. We observed this with other variations in question phrasing and hypothesize that differences in phrasing from the training dataset may be one explanation.
- Finetuned models can save space on the limited context windows of LLMs by incorporating information into the model's parameters, thus reducing the required amount of context. This can leave more room for other information such as conversation history for chat applications. We believe this can be a simpler alternative to ongoing efforts aimed at increasing the context window of LLMs [Zhang et al.(2024)] and adapting RAG to these larger context windows [Xu et al.(2024)].
- Involving end users for testing at various phases of system development can generate feedback that can help steer some of the decision making during development.
- Trees provide an appropriate structure for representing hierarchical information such as entities in an organization that can be used to enhance the context. Our evaluations suggest that this helps make the system fairly robust in responding to questions about entities.

7.1 RAG vs. Finetuning

We used a combination of RAG with finetuning in our application. Both approaches have their strengths and weaknesses. Finetuning requires more computational resources initially to train a model. For smaller applications, the computational requirements of RAG are likely to be minimal beyond the retrieval technologies used. Compared to RAG, Finetuning can allow for adapting the model's writing style and tone to match the organization's document. However, finetuning must be done carefully as updating a model's parameters can degrade its overall language abilities. Testing finetuned models for overfitting, as we have done here, can be a useful way to check for this. While finetuning requires efforts to curate a high quality training dataset and tune hyper-parameters, RAG too requires many optimizations. Looking at the RAG algorithm presented in Table 1, there are many settings that can be explored at each step, including how to chunk the source document, choice of embedding model and retrieval algorithm among others.

Maintaining LLM applications requires updating their knowledge bases as the underlying documents change over time. For RAG, this can be done readily by updating the retrieval database, making it more suitable for dynamic, frequent updates. Updating a finetuned model, however, requires preparing a training dataset and retraining the model, making it more suited to applications such as ours where the underlying document changes less frequently.

Deployed applications receive a diverse variety of user queries. As we observed earlier, finetuned models are still prone to hallucinations especially when faced with unfamiliar input. Using RAG can reduce hallucinations considerably by grounding the model's responses in the given context [Zhang et al.(2023)]. However, RAG also suffers from many limitations across the pipeline and is sensitive to noisy or incomplete contexts resulting in hallucinations and incomplete answers [Barnett et al.(2024)]. Reducing LLM hallucinations is still an open research question and future advances can help make systems more robust as frameworks mature over time [Ye et al.(2023)]. From our experience, hybrid approaches combining RAG and finetuning are likely to be promising for real-world applications and should be explored further.

7.2 Future Work

We received positive feedback on our system from the organization and there is interest to expand the current system to a wider corpus of documents. Another area for future work is to expand the system into a chat-based application. This requires further considerations beyond what is needed for a Q&A application such as effectively handling chat history.

References

[Agrawal et al.(2023a)] Garima Agrawal, Tharindu Kumarage, Zeyad Alghami, and Huan Liu. 2023a. Can Knowledge Graphs Reduce Hallucinations in LLMs? : A Survey. `https://doi.org/10.48550/arXiv.2311.07914` arXiv:2311.07914 [cs].

[Agrawal et al.(2023b)] Garima Agrawal, Kuntal Pal, Yuli Deng, Huan Liu, and Chitta Baral. 2023b. AISecKG: Knowledge Graph Dataset for Cybersecurity Education. In *Proceedings of the AAAI 2023 Spring Symposium on Challenges Requiring the Combination of Machine Learning and Knowledge Engineering (AAAI-MAKE 2023) (CEUR Workshop Proceedings, Vol. 3433)*, Andreas Martin, Hans-Georg Fill, Aurona Gerber, Knut Hinkelmann, Doug Lenat, Reinhard Stolle, and Frank van Harmelen (Eds.). CEUR, Hyatt Regency, San Francisco Airport. `https://ceur-ws.org/Vol-3433/#paper6` ISSN: 1613-0073.

[Baek et al.(2023)] Jinheon Baek, Alham Fikri Aji, and Amir Saffari. 2023. Knowledge-Augmented Language Model Prompting for Zero-Shot Knowledge Graph Question Answering. In *Proceedings of the 1st Workshop on Natural Language Reasoning and Structured Explanations (NLRSE)*, Bhavana Dalvi Mishra, Greg Durrett, Peter Jansen, Danilo Neves Ribeiro, and Jason Wei (Eds.). Association for Computational Linguistics, Toronto, Canada, 78–106. `https://doi.org/10.18653/v1/2023.nlrse-1.7`

[Balaguer et al.(2024)] Angels Balaguer, Vinamra Benara, Renato Luiz de Freitas Cunha, Roberto de M. Estevão Filho, Todd Hendry, Daniel Holstein, Jennifer Marsman, Nick Mecklenburg, Sara Malvar, Leonardo O. Nunes, Rafael Padilha, Morris Sharp, Bruno Silva, Swati Sharma, Vijay Aski, and Ranveer Chandra. 2024. RAG vs Fine-tuning: Pipelines, Tradeoffs, and a Case Study on Agriculture. `https://doi.org/10.48550/arXiv.2401.08406` arXiv:2401.08406 [cs].

[Baldazzi et al.(2023)] Teodoro Baldazzi, Luigi Bellomarini, Stefano Ceri, Andrea Colombo, Andrea Gentili, and Emanuel Sallinger. 2023. Fine-Tuning Large Enterprise Language Models via Ontological Reasoning. In *Rules and Reasoning: 7th International Joint Conference, RuleML+RR 2023, Oslo, Norway, September 18–20, 2023, Proceedings*. Springer-Verlag, Berlin, Heidelberg, 86–94. `https://doi.org/10.1007/978-3-031-45072-3_6`

[Barnett et al.(2024)] Scott Barnett, Stefanus Kurniawan, Srikanth Thudumu, Zach Brannelly, and Mohamed Abdelrazek. 2024. Seven Failure Points When Engineering a Retrieval Augmented Generation System. `http://arxiv.org/abs/2401.05856` arXiv:2401.05856 [cs].

[Chang et al.(2023)] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. 2023. A Survey on Evaluation of Large Language Models. `https://doi.org/10.48550/arXiv.2307.03109` arXiv:2307.03109 [cs].

[Chiang and Lee(2023)] Cheng-Han Chiang and Hung-yi Lee. 2023. Can Large Language Models Be an Alternative to Human Evaluations?. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 15607–15631. `https://doi.org/10.18653/v1/2023.acl-long.870`

[Cuconasu et al.(2024)] Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. 2024. The Power of Noise: Redefining Retrieval for RAG Systems. `https://doi.org/10.48550/arXiv.2401.14887` arXiv:2401.14887 [cs].

[Dettmers et al.(2023)] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient Finetuning of Quantized LLMs. `https://doi.org/10.48550/arXiv.2305.14314` arXiv:2305.14314 [cs].

[Drori et al.(2023)] Iddo Drori, Sarah J. Zhang, Reece Shuttleworth, Sarah Zhang, Keith Tyser, Zad Chin, Pedro Lantigua, Saisamrit Surbehera, Gregory Hunter, Derek Austin, Leonard Tang, Yann Hicke, Sage Simhon, Sathwik Karnik, Darnell Granberry, and Madeleine Udell. 2023. From Human Days to Machine Seconds: Automatically Answering and Generating Machine Learning Final Exams. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23)*. Association for Computing Machinery, New York, NY, USA, 3947–3955. `https://doi.org/10.1145/3580305.3599827`

[Fang et al.(2023)] Chuyu Fang, Chuan Qin, Qi Zhang, Kaichun Yao, Jingshuai Zhang, Hengshu Zhu, Fuzhen Zhuang, and Hui Xiong. 2023. RecruitPro: A Pretrained Language Model with Skill-Aware Prompt Learning for Intelligent Recruitment. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23)*. Association for Computing Machinery, New York, NY, USA, 3991–4002. `https://doi.org/10.1145/3580305.3599894`

[Gao et al.(2024)] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. 2024. Retrieval-Augmented Generation for Large Language Models: A Survey. `https://doi.org/10.48550/arXiv.2312.10997` arXiv:2312.10997 [cs].

[Guo et al.(2022)] Quan Guo, Shuai Cao, and Zhang Yi. 2022. A medical question answering system using large language models and knowledge graphs. *International Journal of Intelligent Systems* 37, 11 (2022), 8548–8564. `https://doi.org/10.1002/int.22955` _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/int.22955.

[Guu et al.(2020a)] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020a. Retrieval Augmented Language Model Pre-Training. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 3929–3938. `https://proceedings.mlr.press/v119/guu20a.html` ISSN: 2640-3498.

[Guu et al.(2020b)] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020b. REALM: retrieval-augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning (ICML'20, Vol. 119)*. JMLR.org, 3929–3938.

[Gómez-Rodríguez and Williams(2023)] Carlos Gómez-Rodríguez and Paul Williams. 2023. A Confederacy of Models: a Comprehensive Evaluation of LLMs on Creative Writing. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 14504–14528. `https://doi.org/10.18653/v1/2023.findings-emnlp.966`

[Hamidi and Roberts(2023)] Alaleh Hamidi and Kirk Roberts. 2023. Evaluation of AI Chatbots for Patient-Specific EHR Questions. `https://doi.org/10.48550/arXiv.2306.02549` arXiv:2306.02549 [cs].

[He et al.(2023)] Kai He, Rui Mao, Qika Lin, Yucheng Ruan, Xiang Lan, Mengling Feng, and Erik Cambria. 2023. A Survey of Large Language Models for Healthcare: from Data, Technology, and Applications to Accountability and Ethics. `https://doi.org/10.48550/arXiv.2310.05694` arXiv:2310.05694 [cs].

[Hendrycks et al.(2021)] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring Massive Multitask Language Understanding. `https://doi.org/10.48550/arXiv.2009.03300` arXiv:2009.03300 [cs].

[Houlsby et al.(2019)] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-Efficient Transfer Learning for NLP.

In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 2790–2799. `https://proceedings.mlr.press/v97/houlsby19a.html`

[Howard and Ruder(2018)] Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Iryna Gurevych and Yusuke Miyao (Eds.). Association for Computational Linguistics, Melbourne, Australia, 328–339. `https://doi.org/10.18653/v1/P18-1031`

[Hu et al.(2021)] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. `https://doi.org/10.48550/arXiv.2106.09685` arXiv:2106.09685 [cs].

[Huang et al.(2023)] Allen H. Huang, Hui Wang, and Yi Yang. 2023. FinBERT: A Large Language Model for Extracting Information from Financial Text*. *Contemporary Accounting Research* 40, 2 (2023), 806–841. `https://doi.org/10.1111/1911-3846.12832` _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/1911-3846.12832.

[Huang et al.(2022)] Jizhou Huang, Haifeng Wang, Yibo Sun, Yunsheng Shi, Zhengjie Huang, An Zhuo, and Shikun Feng. 2022. ERNIE-GeoL: A Geography-and-Language Pre-trained Model and its Applications in Baidu Maps. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22)*. Association for Computing Machinery, New York, NY, USA, 3029–3039. `https://doi.org/10.1145/3534678.3539021`

[Kandpal et al.(2023)] Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large language models struggle to learn long-tail knowledge. In *Proceedings of the 40th International Conference on Machine Learning (ICML'23, Vol. 202)*. JMLR.org, Honolulu, Hawaii, USA, 15696–15707.

[Lai et al.(2023)] Tin Lai, Yukun Shi, Zicong Du, Jiajie Wu, Ken Fu, Yichao Dou, and Ziqi Wang. 2023. Psy-LLM: Scaling up Global Mental Health Psychological Services with AI-based Large Language Models. `https://doi.org/10.48550/arXiv.2307.11991` arXiv:2307.11991 [cs].

[Lewis et al.(2020)] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems*, Vol. 33. Curran Associates, Inc., 9459–9474. `https://proceedings.neurips.cc/paper_files/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html`

[Lialin et al.(2023)] Vladislav Lialin, Vijeta Deshpande, and Anna Rumshisky. 2023. Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning. `https://doi.org/10.48550/arXiv.2303.15647`

[Liu et al.(2023)] Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment. `https://doi.org/10.48550/arXiv.2303.16634` arXiv:2303.16634 [cs].

[Liventsev et al.(2023)] Vadim Liventsev, Anastasiia Grishina, Aki Härmä, and Leon Moonen. 2023. Fully Autonomous Programming with Large Language Models. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '23)*. Association for Computing Machinery, New York, NY, USA, 1146–1155. `https://doi.org/10.1145/3583131.3590481`

[Louis et al.(2023)] Antoine Louis, Gijs van Dijck, and Gerasimos Spanakis. 2023. Interpretable Long-Form Legal Question Answering with Retrieval-Augmented Large Language Models. `https://doi.org/10.48550/arXiv.2309.17050` arXiv:2309.17050 [cs].

[Min et al.(2017)] Sewon Min, Minjoon Seo, and Hannaneh Hajishirzi. 2017. Question Answering through Transfer Learning from Large Fine-grained Supervision Data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Regina Barzilay and Min-Yen Kan (Eds.). Association for Computational Linguistics, Vancouver, Canada, 510–517. `https://doi.org/10.18653/v1/P17-2081`

[Nellis and Cherney(2023)] Stephen Nellis and Max A. Cherney. 2023. US curbs AI chip exports from Nvidia and AMD to some Middle East countries. *Reuters* (Aug. 2023). `https://www.reuters.com/technology/us-restricts-exports-some-nvidia-chips-middle-east-countries-filing-2023-08-30/`

[OpenAI et al.(2023)] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mo Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek

Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, C. J. Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2023. GPT-4 Technical Report. `https://doi.org/10.48550/arXiv.2303.08774` arXiv:2303.08774 [cs].

[Ram et al.(2023)] Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-Context Retrieval-Augmented Language Models. `https://doi.org/10.48550/arXiv.2302.00083` arXiv:2302.00083 [cs].

[Su et al.(2023)] Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. 2023. One Embedder, Any Task: Instruction-Finetuned Text Embeddings. In *Findings of the Association for Computational Linguistics: ACL 2023*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 1102–1121. `https://doi.org/10.18653/v1/2023.findings-acl.71`

[Touvron et al.(2023)] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. `https://doi.org/10.48550/arXiv.2307.09288` arXiv:2307.09288 [cs].

[Vaswani et al.(2017)] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information*

Processing Systems, Vol. 30. Curran Associates, Inc. `https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html`

[Wang et al.(2023)] Yidong Wang, Zhuohao Yu, Zhengran Zeng, Linyi Yang, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, Xing Xie, Wei Ye, Shikun Zhang, and Yue Zhang. 2023. PandaLM: An Automatic Evaluation Benchmark for LLM Instruction Tuning Optimization. `https://doi.org/10.48550/arXiv.2306.05087` arXiv:2306.05087 [cs].

[Wu et al.(2023)] Yike Wu, Nan Hu, Sheng Bi, Guilin Qi, Jie Ren, Anhuan Xie, and Wei Song. 2023. Retrieve-Rewrite-Answer: A KG-to-Text Enhanced LLMs Framework for Knowledge Graph Question Answering. `https://doi.org/10.48550/arXiv.2309.11206` arXiv:2309.11206 [cs].

[Xia et al.(2022)] Fei Xia, Bin Li, Yixuan Weng, Shizhu He, Kang Liu, Bin Sun, Shutao Li, and Jun Zhao. 2022. MedConQA: Medical Conversational Question Answering System based on Knowledge Graphs. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Wanxiang Che and Ekaterina Shutova (Eds.). Association for Computational Linguistics, Abu Dhabi, UAE, 148–158. `https://doi.org/10.18653/v1/2022.emnlp-demos.15`

[Xiao et al.(2022)] Shitao Xiao, Zheng Liu, Yingxia Shao, Tao Di, Bhuvan Middha, Fangzhao Wu, and Xing Xie. 2022. Training Large-Scale News Recommenders with Pretrained Language Models in the Loop. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22)*. Association for Computing Machinery, New York, NY, USA, 4215–4225. `https://doi.org/10.1145/3534678.3539120`

[Xu et al.(2023)] Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. 2023. Parameter-Efficient Fine-Tuning Methods for Pretrained Language Models: A Critical Review and Assessment. `https://doi.org/10.48550/arXiv.2312.12148` arXiv:2312.12148 [cs].

[Xu et al.(2024)] Peng Xu, Wei Ping, Xianchao Wu, Lawrence McAfee, Chen Zhu, Zihan Liu, Sandeep Subramanian, Evelina Bakhturina, Mohammad Shoeybi, and Bryan Catanzaro. 2024. Retrieval meets Long Context Large Language Models. `https://doi.org/10.48550/arXiv.2310.03025` arXiv:2310.03025 [cs].

[Yang et al.(2023)] Fangkai Yang, Pu Zhao, Zezhong Wang, Lu Wang, Jue Zhang, Mohit Garg, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. 2023. Empower Large Language Model to Perform Better on Industrial Domain-Specific Question Answering. `http://arxiv.org/abs/2305.11541` arXiv:2305.11541 [cs].

[Ye et al.(2023)] Hongbin Ye, Tong Liu, Aijia Zhang, Wei Hua, and Weiqiang Jia. 2023. Cognitive Mirage: A Review of Hallucinations in Large Language Models. `https://doi.org/10.48550/arXiv.2309.06794` arXiv:2309.06794 [cs].

[Zhang et al.(2024)] Yikai Zhang, Junlong Li, and Pengfei Liu. 2024. Extending LLMs' Context Window with 100 Samples. `https://doi.org/10.48550/arXiv.2401.07004` arXiv:2401.07004 [cs].

[Zhang et al.(2023)] Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. 2023. Siren's Song in the AI Ocean: A Survey on Hallucination in Large Language Models. `https://doi.org/10.48550/arXiv.2309.01219` arXiv:2309.01219 [cs].

[Zhao et al.(2023)] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A Survey of Large Language Models. `https://doi.org/10.48550/arXiv.2303.18223` arXiv:2303.18223 [cs].

A Prompts

A.1 Prompts for Question and Answer Instruction Dataset Generation

PROMPT1:

As an AI assistant, your task is To generate question types, questions, answers, and source text for answers that comprehensively cover a given text and present them in a table format, considering various question types such as dialogs, conversations, conversations summary, conversations fact check, fact check, True or False, summary, quiz, review, Fill in the Blank, Short Answer, Yes or No, Matrix, compound questions, etc., follow these steps:

1. Read the given text: Carefully read the text to understand its content and context.
2. Identify key information: Look for definitions, metrics, facts, figures, events, or statements that can be used to create questions, ensuring that all parts of the text are covered.
3. Determine question types: Based on the key information, decide which question types would be most suitable for the text (dialogs, conversations, conversations summary, conversations fact check, fact check, True or False, summary, quiz, review, Fill in the Blank, Short Answer, Yes or No, Matrix, compound questions, classify question, etc.).
4. Generate questions: Create questions according to the chosen question types, ensuring they are clear, concise, and aligned with the question type. Make sure the questions cover the entire text.
5. Locate answers: Find the corresponding answers for the questions within the text.
6. Obtain source text: Replicate the essential text sections that offer the solutions to the queries, functioning as the basis for the responses. The source text must be thoroughly exhibited in a complete manner. It is necessary for the source text to be presented in a clear and understandable English without utilizing ellipsis.
7. Design a table: Create a table with columns for question type, question, answer, and source text.
8. Populate the table: Fill the table with the generated question types, questions, answers, and the extracted source text.
9. Review and revise: Double-check the table for accuracy, comprehensiveness, and make any necessary adjustments.

Here’s an example of how to structure your table:

| Question Type | Question | Answer | Source Text |

|——————————|————-|———|—————————-|

| Dialog | Q1 | A1 | Source Text 1 |

| Conversation | Q2 | A2 | Source Text 2 |

| Conversation Summary | Q3 | A3 | Source Text 3 |

| Conversation Fact Check| Q4 | A4 | Source Text 4 |

| Fact Check | Q5 | A5 | Source Text 5 |

| True or False | Q6 | A6 | Source Text 6 |

| Summary | Q7 | A7 | Source Text 7 |

| Quiz | Q8 | A8 | Source Text 8 |

| Review | Q9 | A9 | Source Text 9 |

| Fill in the Blank | Q10 | A10 | Source Text 10 |

| Short Answer | Q11 | A11 | Source Text 11 |

| Yes or No | Q12 | A12 | Source Text 12 |

| Matrix | Q13 | A13 | Source Text 13 |

| Compound Question | Q14 | A14 | Source Text 14 |

| ... | ... | ... | ... |

PROMPT2:

As an AI assistant, your task is To generate a Q&A conversation between a user and an assistant using a given text, follow these steps:

1. Read and understand the text: Carefully read the given text and make sure you understand the key points, facts, and ideas presented in it.
2. Identify potential questions: Based on your understanding of the text, think of possible questions that a user might ask related to the content. Focus on questions that cover the main ideas and details from the text. Try to come up with at least 2-3 questions per topic.
3. Formulate clear and concise answers: For each question, draft a clear and concise answer based on the information provided in the text. Make sure your answers are accurate, informative, and directly address the questions.
4. Organize the conversation: Arrange the questions and answers into a logical order, creating a smooth flow for the conversation. Use a back-and-forth format where the user asks a question and the AI assistant provides an answer, followed by another question and answer, and so on. Ensure that there are at least 2-3 exchanges per topic.
5. Check for clarity and coherence: Review the conversation to ensure that it is clear, coherent, and easy to follow. Make sure that the questions and answers are related to the given text and accurately represent the content.

Here's an example using a sample text about the benefits of a healthy diet:

Text: "A healthy diet consists of various food groups, such as fruits, vegetables, whole grains, lean proteins, and healthy fats. Consuming a balanced diet can help maintain a healthy weight, support the immune system, and reduce the risk of chronic diseases."

Conversation:

User: What are the main food groups in a healthy diet?

Assistant: A healthy diet includes various food groups, such as fruits, vegetables, whole grains, lean proteins, and healthy fats.

User: How can a healthy diet help maintain a healthy weight?

Assistant: A balanced diet provides the essential nutrients your body needs without excessive calories. Consuming the right balance of foods can help you achieve and maintain a healthy weight by preventing overeating and ensuring proper energy intake.

User: What are the benefits of a healthy diet for the immune system?

Assistant: A healthy diet supports the immune system by providing essential vitamins, minerals, and antioxidants that help the body fight off infections and maintain overall health.

User: How does a balanced diet reduce the risk of chronic diseases?

Assistant: Consuming a balanced diet can lower the risk of chronic diseases by providing the necessary nutrients for optimal body function and reducing the intake of harmful substances like saturated fats, trans fats, and added sugars. A healthy diet can help prevent conditions such as heart disease, diabetes, and certain types of cancer.

PROMPT3:

As an AI assistant, your task is To generate question types, questions, answers, and source text for answers that comprehensively cover a given text and sample questions present them in a table format, follow these steps:

1. Read the given text: Carefully read the text to understand its content and context.

2. Identify key information: Look for definitions, metrics, facts, figures, events, or statements that can be used to create questions, ensuring that all parts of the text are covered.
3. Generate questions: Create questions according to the chosen question types, ensuring they are clear, concise, and aligned with the question type. Make sure the questions cover the entire text.
4. Locate answers: Find the corresponding answers for the questions within the text.
5. Obtain source text: Replicate the essential text sections that offer the solutions to the queries, functioning as the basis for the responses. The source text must be thoroughly exhibited in a complete manner. It is necessary for the source text to be presented in a clear and understandable English without utilizing ellipsis.
6. Design a table: Create a table with columns for question type, question, answer, and source text.
7. Populate the table: Fill the table with the generated question types, questions, answers, and the extracted source text.
8. Review and revise: Double-check the table for accuracy, comprehensiveness, and make any necessary adjustments.

Here's Sample Questions:

- Can you describe the responsibilities of the <committee name> and the <committee name> under the <entity name>?
- What are the categories of <entity group name>, and could you provide examples of entities falling under these categories?
- Who are the stakeholders that must abide by the principles outlined in the Governance Manual, according to the text?
- Can you provide a comprehensive list of all the authorized documents that are recognized by the Governance Manual and contribute to its overall objectives and goals?
- Where is <entity name> in the Governance Framework?
- What is <entity name>?
- What are the entities within the <category name> of the Framework?

Here's an example of how to structure your table:

| Question Type | Question | Answer | Source Text |

|——————————|————-|———-|————————-|

| Dialog | Q1 | A1 | Source Text 1 |

| Conversation | Q2 | A2 | Source Text 2 |

| Conversation Summary | Q3 | A3 | Source Text 3 |

| Conversation Fact Check| Q4 | A4 | Source Text 4 |

| Fact Check | Q5 | A5 | Source Text 5 |

| True or False | Q6 | A6 | Source Text 6 |

| Summary | Q7 | A7 | Source Text 7 |

| Quiz | Q8 | A8 | Source Text 8 |

| Review | Q9 | A9 | Source Text 9 |

| Fill in the Blank | Q10 | A10 | Source Text 10 |

| Short Answer | Q11 | A11 | Source Text 11 |

| Yes or No | Q12 | A12 | Source Text 12 |

| Matrix | Q13 | A13 | Source Text 13 |

| Compound Question | Q14 | A14 | Source Text 14 |

| ... | ... | ... | ... |

Remember to generate a variety of question types, questions, answers, and source text, and aim for a total of 10 questions or more questions.

A.2 System Prompt for inference

As a <organization document name> assistant, Your job is provide with accurate information exclusively based on the <organization document name> and utilizing the given context. Only use the context if it is relevant to the question.

Follow the below instructions to answer questions in a step-by-step manner:

1. Carefully analyze the question to comprehend what is being asked. Pay attention to the keywords, indirect cues, intentions, and specific details mentioned in the question.
2. Find relevant information within the <organization document name> and provided context to construct your answer. If the given context is not suitable, do not use it.
3. Support your answer with facts or information from the <organization document name> and provided context. If the given context is not relevant, do not need to use it.
4. Organize the main points in a concise and clear manner.
5. DO NOT create answers that aren’t in the <organization document name>.
6. Provide brief and clear response to the question, focusing on main point ONLY.
7. If you can’t answer a question, politely state that you’re unable to do so.

Please take note that <organization acronym> and <organization name> have identical meanings. Additionally, phrases such as "types", "departments", "entities", "categories", "positions", "groups", "units", "classifications" and "directorates" might also suggest the same idea. Furthermore, any significant keywords like "All" or "comprehensive" imply that the information data is sourced only from the manual.

Remember, as an AI assistant, it is vital to provide precise and dependable information based on the <organization document name>.

B Code Snippets for Entity Tree

This sections shows the code for defining the entity parser component in the spaCy library and the code for parsing entities from user queries and performing the tree search.

Code Snippet 1:

Defining a rule-based entity name matching rule in spaCy.

```
def enhance_spaCY(ent_name):
    nlp = spacy.load("en_core_web_sm")
    ruler = nlp.add_pipe("span_ruler")

    patterns=[]

    for t in ent_name:
        temp = t.split()
        if len(temp) > 1 and t.find(',')<0 and t.find('&')<0:

            temp_pattern = []
            for a_temp in temp:
                temp_pattern.append({"LOWER": a_temp.lower()})
            key_id = find_key_entity(t)

            patterns.append({"label": "UNHCRORG", "pattern": temp_pattern,
                             "id": key_id.lower()})
        else:
            key_id = find_key_entity(t)

            patterns.append(
                {"label": "UNHCRORG", "pattern": t.lower(),
                "id": key_id.lower()})

    ruler.add_patterns(patterns)

    return nlp
```

Code Snippet 2:

Entity parsing from the user's query. spaCy library extracts named entities based on the rule defined earlier. The tree context can then be generated for the entities mentioned in the user's query and returned as context information.

```
def search_entity_info(tree, nlp, search):
    search_context=[]
    search = search.lower().strip()

    doc = nlp(search)
    for span in doc.spans["ruler"]:
        if span.label_ == 'UNHCRORG':
            search_context.append(get_node_info(tree, span.id_))

    return '\n'.join(search_context)
```