RAG (Retrieval Augmented Generation) for Enhanced Content Creation

Hongyan Yang

Abstract

In this work, I developed a system that leverages Retrieval-Augmented Generation (RAG) for the creation of precise and contextually appropriate responses within the data risk management domain. The whole project was developed from the ground up with custom functions to efficiently extract domain content, augment user's prompt and to generate answers. Different text generation models are prompt engineered or fine-tuned and their performance is evaluated with suitable metrics. This project demonstrates the application of RAG in specialized domains and analyzes various approaches to generate answers. (Click to visit the Repository)

Introduction

Modern enterprises struggle to answer user's query regarding company polices, legal requirement or guidance in an efficient and timely manner. This project is motivated by this challenge and aims to provide a solution to address it. Conventional user support approach and generative language models always fail for the following reasons:

- Company policy and guidance is of large volume and requires careful version management and updates.
- Traditional search engines that are based on key word search often fall short when users provide imprecise key word of their queries.
- Output of traditional question-answer system that composed of links to relevant documents cannot satisfy users with concise and explanatory answer.
- Generative language models are prone to providing contextually inappropriate or incorrect content and suffer from hallucination.

In this work, a question-answer system utilizing RAG framework is designed to conquer the above issues. It enhances the quality and relevance of responses with self-defined functions and algorithms and integrates key features of RAG, which include:

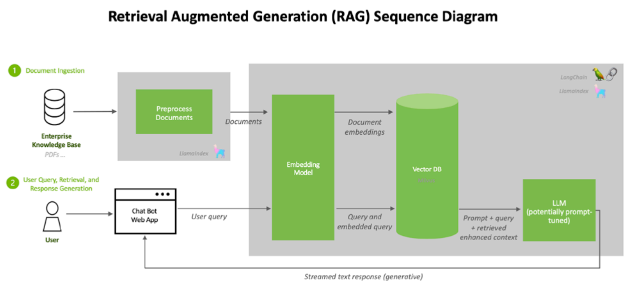
- Extract and store domain-specific knowledge into local base by efficiently constructing and converting document chunks that enables rapid retrieval.
- Augment user's query by finding relevant chunks in the knowledge base via similarity search to form the context with specificity and depth.
- Generate answers to the query with a fine-tuned encoder-decoder seq2seq model. The model is trained with suitable dataset to cultivate the abstractive question-answer ability.
- Generate answers with a prompt-engineered decoder-only casual language model. The model is adapted to answer in the required format after understanding relevant context.
- The above two approaches to generate answers are evaluated quantitatively with corresponding metrics and their potential are analyzed.

The above two approaches to answer generation are assessed using quantitative metrics. Their strengths and potential limitations in practical application are analyzed. Overall, the system leverages RAG to provide users with accurate, concise, and informative answers to inquiries related to policies, legal documentation, and procedural guidelines in the working environment.

Background

The system is built off a RAG workflow, which combines both offline and online stages as shown in Figure 1. The ingestion of domain-knowledge occurs offline while the user's query input, query augmentation and answer output are processed online in real time.

The offline part of the workflow deals with document preprocessing and embedding vectors generation, while the online part takes charge of query understanding and answer



Retrieval Augmented Generation (RAG) Sequence Diagram
1 Document Ingestion
Enterprise Knowledge Base
PDFs ...
Preprocess Documents
LlamaIndex
Documents
LangChain
LlamaIndex
Embedding Model
Document embeddings
Vector DB
2 User Query, Retrieval, and Response Generation
User
Chat Bot Web App
User query
Query and embedded query
Prompt + query + retrieved enhanced context
LLM (potentially prompt-tuned)
Streamed text response (generative)

Figure 1: Overview of RAG pipeline components: ingest and query flows

generation. These two parts interwork with each other and a brief description of the key functions are as follows:

- Document pre-processing: raw documents from specific domain are processed to create chunks of sentences. Each chunk is of similar length so less information will be lost after it being tokenized and vectorized.
- Embedding generation: A language embedding model is used to transform both chunks of documents and user's query from text into high-dimensional semantic vectors.
- Query augmentation: The RAG system retrieves relevant chunks of knowledge by calculating the similarity between user's query vector and chunk vectors. The top k relevant chunks will be used to answer the query accurately in the later stage.
- Answer generation: Generative language models are used to output answer to user's query by understanding the query and leveraging the given context sentences.

The system incorporates the advantage of RAG from the above stages with custom functions and methods. Different approaches are designed and analyzed in the Document pre-processing stage and the Answer generation stage.

Related Work

Document Retrieval

Prior work to tackle open-domain question answering has shown the effectiveness of combining knowledge source to generate responses. Also, an efficient search method to find the relevant documents is the core to perform this task. [1] In this project, I designed a fast way to extract and store domain documents via indexed data frame and matrix which enables secure local file management and efficient knowledge retrieval.

Transfer Learning

Prior work illustrates a useful technique in natural language processing that a model pre-trained on a data-rich task has the potential to be fine-tuned on other downstream tasks. [2] Prior work also shows that instruction finetuning is a general approach to enhancing the quality and usability of large language models. [3] In this project, flan-t5-base, a seq2seq model is instruction finetuned with the SQuAD dataset to train the abstractive question answer ability. The fine-tuned model's performance is evaluated quantitatively with corresponding metrics.

Prompt Engineering

Other studies analyzed the role of prompt engineering to optimize the form of output of language models and to get full use of models' capabilities. [4] In this project, Gemma-7b, a text generation model is fed with prompt in pre-defined format to output response to user's augmented query in desired manner.

Text Generation

Recent studies also demonstrate language models' strong ability to comprehend language and generate answers. [5] The Gemma-7b model is applied to output grammar-correct and context-coherent answers to user's queries after understanding the domain knowledge inherent in the context. Chunks of relevant documents that are retrieved are often separated from each other, so a language model capable of fully digesting the sentences is of good importance.

RAG Models

This project follows the framework of RAG models, which shows its promise in enhancing response quality by combining pre-trained parametric and non-parametric memory for language generation. [6] There are two differences between my model and the original RAG model:

- Instead of splitting documents into disjoint 100-word chunks, custom functions are applied in this project to create chunks with same number of sentences. Also, users can choose the number of repeated sentences among consecutive chunks. These modifications are made to avoid splitting tokens in a complete sentence and losing embedded information in the context.
- Other than simply concatenating input with the retrieved content, my model prepends instruction to the prompt before feeding them to the instruction finetuned language model. Thus, the answers follow a desired format and manner.

Project Description

This project follows RAG models' pipeline to create enhanced content. Components of the system and related functions and equations are as follows:

Step 1. Retrieve Domain Knowledge

Domain raw documents mainly in the form of pdf are stored through local path for better maintenance and future scaling. The information is designed to be retrieved page by page so the model response can be traced back to a certain page of the raw document.

By using Algorithm 1, each page is extracted as a dictionary because the key-value pair can be easily converted to pandas dataframe for fast searching and easy indexing in later stages.

```
Algorithm 1: Document Extraction
Input: The path to the raw documents
Output: A list of page dictionaries
1:  Let output-list = list().
2:  For document in input path:
3:      Read the document and get content page by page.
4:      Conduct basic string formatting
5:      output-list.append({'title': title,
                            'page number': index,
                            'content': content})
6:  return output-list
```

```
Algorithm 2: Chunks Creation
Input: A list of page dictionaries
Parameter: chunk-size, overlapped-number
Output: A list of page dictionaries with chunks
1:  Let output-list = list().
2:  For page in input-list:
3:      page-content = page['content']
4:      sentences-list = [page-content's sentences]
5:      For sentences in sentences-list:
6:          Try to concatenate sentence of length chunk-size
7:          if len(chunk) < chunk-size then
8:              Get sentences from the next page and record
                the number.
9:              Chunks from the next page are formed from
                new sentences given overlapped number.
10:         else
11:             Next chunk is created given overlapped num-
                ber to the previous chunk.
12:         end if
13:     output-list.append({'title': title,
                            'page number': index,
                            'content': content,
                            'chunks': chunks})
14: return output-list
```

Another custom algorithm (Algorithm 2) is applied to create chunks of sentences out of each page after page paragraphs are extracted and stored in a structured way. The algorithm aims at making chunks of the same number of sentences so that each chunk is optimized to represent a similar amount of information. A parameter of 'overlapped number' is also introduced to control the number of repeated sentences between two consecutive chunks, which eases the later semantic similarity search by controlling the correlation between consecutive chunks. It also helps to retain the knowledge that may be conveyed through more than one chunk. Chunks from one page may have fewer sentences than 'chunk size', in this case, sentences from the next page are used to fill the chunk to expected length. At last, chunks are added to the page's dictionary to be indexed and structured.

The last step to construct the knowledge base is transforming chunks from text strings to numeric semantic vectors. An appropriate sentence embedding model, 'all-mpnet-base-v2', is used to complete this task. The same model is also used to convert user's query to vectors in the same hyper space. So, relevant chunks can be retrieved through similarity search by computing the inner product of query's vector and chunk vectors.

Step 2. Prompt Augmentation

User's query is augmented by a context composed of relevant chunks before feeding the generative language model. Top k related chunks are selected via similarity search using Algorithm 3 and are returned as a list of strings.

```
Algorithm 3: Get Top Chunks
Input: A string of user's query, top-k
Parameter: chunk-vectors, chunk-df, embedding-model
Output: A list of top-k relevant chunks
1: Embed the query with the embedding model.
2: # Get the similarity matrix with dot product.
3: similarity-mat = dot-score(query-vector, chunk-vectors)
4: # Get the top-k indices
5: top-indices = topk(similarity-mat, k= top-k)
6: top-chunks = chunk-df[top-indices]['chunk-string']
7: return top-chunks.tolist()
```

Step 3. Generative Model Selection and Training

Two models are selected and trained in varying ways to generate answers.

The first model 'google/flan-t5-base', an encoder-decoder seq2seq model, is instruction finetuned with the SQuAD dataset to train its abstractive question answer ability.

The narrativeqa dataset is also tested to train the model but failed because the contexts of narrativeqa dataset are long documents designed for reading comprehension. Although the open questions and abstractive human answers match the use case of this project, the long context of narrativeqa with an average of 200,000 tokens is beyond flan-t5-base's maximum encoder length of 1024. Additionally, the long context is not compatible with RAG's context of under 1000 tokens for top-10 chunks.

The 'google/flan-t5-base' model is trained on SQuAD dataset's samples with self-defined instructions and the desired outputs as shown in Algorithm 4.

A LoRA adapter of alpha=32 is implemented to train the model with relatively small number of additional trainable parameters. After 1.41% of trainable weights training for 3 epochs, the model is adapted to follow user's instruction to generate abstractive answers by leveraging chunks of context in a concise manner. Compared with original model's zero-shot output of simply printing out the last paragraph of the chunk, the finetuned model is more responsive and accurate in executing user's command.

The second model 'google/gemma-7b-it' is a 7B instruct version of the Gemma model which is decoder-only and well capable of text generation tasks, including question an-

```
Algorithm 4: Instruction Finetuning flan-t5-base model
Input: base model,
       dataset's sample context, query, and answers
Parameter: LoRA adapter arguments
Output: fine-tuned model
1: Preprocess the dataset to extract needed parts.
2: Tokenize the question-answer pairs into explicit in-
   structions for the generation model.
3: Setup the LoRA model for fine-tuning.
4: Create a Seq2SeqTrainer instance and train the model
5: return fine-tuned model
```

swering. The model is prompt engineered with input in predefined format to answer user's query in desired manner and ideal style.

The Gemma model is also optimized to be compatible with Flash Attention 2 [7], a technique to speedup calculations across attention layers with better partition and parallelism of GPU works, which makes the model possible to deploy locally with limited resources and suitable to this project.

An instruction prompt is created for the model with few-shot inference. The model is required to answer user's query with information from given context and is guided to first extract relevant passages from the context before responding. Three pairs of question and answer are shown as examples to the model, which illustrates the ideal answer style in an informative and concise manner. Finally, the relevant context extracted from the previous steps are incorporated to the prompt with user's query. The model is instructed to extract relevant passages from the context in the first place.

The language model answers query precisely with parameter 'temperature' set to 0.7, a number lower than 1 that makes the model generate words with less creativity. By sharpening the softmax output distribution, the model produces more conservative and deterministic outputs.

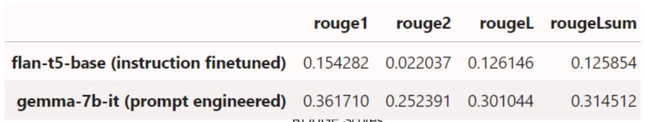
Compared with original model's output in the form only general information that is biased to hallucination, the prompt-engineered model response to domain-specific questions such as query to regulator policies with accurate and meaningful information.

Empirical Results

To evaluate models' performance in a quantitative way, twelve test questions are created specifically in the data risk management domain with human experts' answers attached to each question. Normal question answering datasets are not used because this project's document base is composed of domain-specific policy, legal documents, and guidebooks, which are not compatible with general questions.

Two metrics are applied for models' comparison:

- ROUGE metric is used to assess the quality of answers by comparing them against reference texts created by human experts. As Figure 2 shows, the prompt-engineered gemma-7b-it model outperformed the instruction-fine-tuned flan-t5-base model because the former model has more overlaps between n-grams in the candidate answers and those in reference answers. It implies that the former model is more effective in capturing the content and meaning of the context chunks to answer user's query.
- BLEU metric is also applied for models' assessment because it focuses on the precision of the outcome and is suitable for measuring the output accuracy of the RAG model. As Figure 3 illustrates, the prompt-engineered 'gemma-7b-it' model outperformed the instruction-fine-tuned flan-t5-base model because the former model's answers have higher precision by accurately conveying output with domain knowledge.



	rouge1	rouge2	rougeL	rougeLsum
flan-t5-base (instruction finetuned)	0.154282	0.022037	0.126146	0.125854
gemma-7b-it (prompt engineered)	0.361710	0.252391	0.301044	0.314512

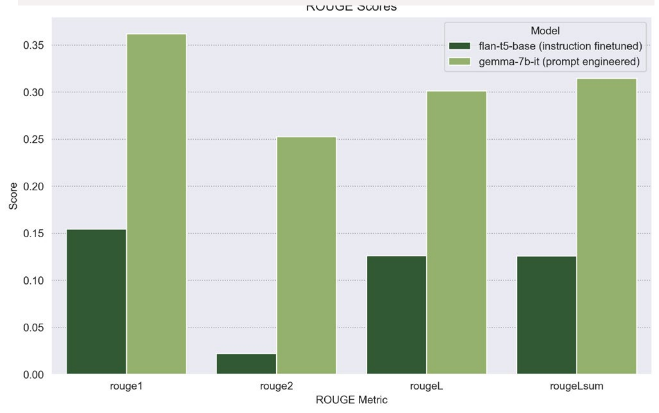
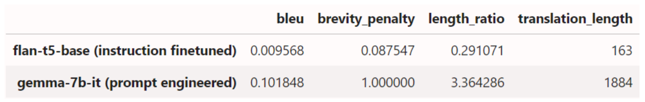
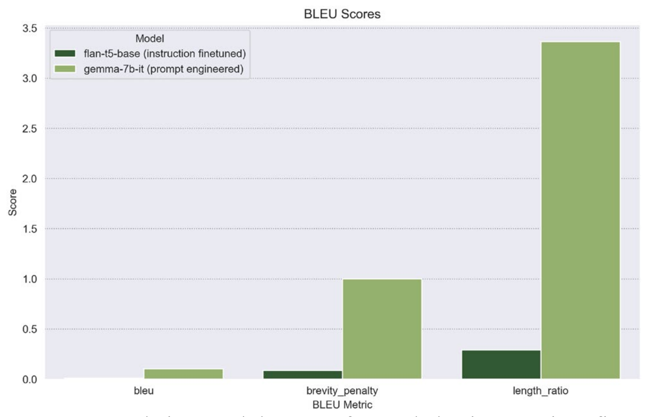


Figure 2: ROUGE Scores of two models

Figure 3: BLEU Scores of two models



	bleu	brevity_penalty	length_ratio	translation_length
flan-t5-base (instruction finetuned)	0.009568	0.087547	0.291071	163
gemma-7b-it (prompt engineered)	0.101848	1.000000	3.364286	1884



Although it takes a long time to instruction finetune the 'flan-t5-base' model, it is surprising that it underperforms the prompt-engineered 'gemma-7b-it' model as assessed by both metrics. After analyzing the training process and the form of the answers that models generate, the model failed for the following reasons:

- The dataset SQuAD that the 'flan-t5-base' trained on is originally designed for extractive question answering, and the labels of the dataset are not complete human sentences but parts of original context. So, the fine-tuned 'flan-t5-base' model tends to provide limited short answers rather than explanatory and informative long answers after understanding the context.

For example, when asked the question ‘What is data identification as one key activity of data governance?’, the fine-tuned ‘flan-t5-base’ model answers ‘Data Governance Boards, Data Councils, or Data Strategy Teams’. In this case, the model correctly finds the subject to take the action but does not provide more details.

In comparison, instead of just picking phrases of context that mostly related to the question, the prompt-engineered ‘gemma-7b-it’ model answers ‘Data identification is one key activity of data governance as defined in the Data Governance Playbook. It involves identifying data assets and creating a data inventory with appropriate metadata.’. This answer is far closer to and even better than the human expert answer ‘Identify data assets and develop a data inventory with appropriate metadata.’ Because it covers the definition of ‘data identification’ in human’s answer and points out that the data source is ‘Data Governance Playbook’.

In summary, a better dataset designed specifically for abstractive question answering is needed to instruction finetune the encoder-decoder seq2seq model.

- The raw documents that form the domain base are of limited number in this project. With only eight pdf files with an average page number of fifty-seven, the encoder-decoder model has limited quality context chunks to understand and aggregate the information. In comparison, lacking ample documents has a smaller impact on the decoder-only model because it is more capable of generating next tokens even outside domain knowledge.
- The gemma-7b model has a larger size than flan-t5 model, which indicates a strong ability to comprehend and generate answers.

Broader Implications

This project builds and implements the RAG model from scratch to create a system aiming at enhancing the content creation for question answering in company’s working environment. This work focuses on providing accurate answers to data risk management policies, legal documents, and government guidebooks. Other work cases can be adapted by setting up a different document source.

An efficient knowledge base construction method is used to manage the semantic vectors in a secure way with possible scaling in a real-life use case.

Two different language models to generate answers are tested and compared, which shows the potential of large models to deploy in situations with limited resources such as company’s own cloud platform. It also shows that state-of-the-art AI models can be easily accessed and broadly applied to everyone’s daily work. With good access authorization and domain file management, the system can offer professional answers in a concise and accurate manner without privacy or legal concerns.

Conclusions

Two different language models are trained and evaluated to perform the answer generation task. The gemma-7b-it model outperforms the fine-tuned flan-t5-base model because of its better language generation ability. However, if the latter model is trained with better datasets with more human expert’s answers, the light-weighted t5 model may perform better.

In the future, a user interface can be added to interact with users. The system in this project can function as the backbone of a Chatbot that can be easily integrated into a company’s daily work through the Team platform.

Please find the link to the GitHub Repository as follows: https://github.com/HongyanYang825/Large-language-models

References

[1] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to Answer Open-Domain Questions. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

[2] Zhuang, F. 2020. A Comprehensive Survey on Transfer Learning. arXiv:1911.02685.

[3] Chung, H. 2022. Scaling Instruction-Finetuned Language Models. arXiv:2210.11416.

[4] Chen, B. 2023. Unleashing the potential of prompt engineering in Large Language Models: a comprehensive review. arXiv:2310.14735.

[5] Mesnard, T. 2024. Gemma: Open Models Based on Gemini Research and Technology. arXiv:2403.08295.

[6] Lewis, P. 2021. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. arXiv:2005.11401.

[7] Dao, T. 2023. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. arXiv:2307.08691.