

Custom Knowledge Retrieval Pipeline with LLMs for Question Answering Tasks

Marco Wang | ShaoBo Zhang

Motivation

We both wanted to experiment with Large Language Models (LLMs) and their capabilities to interact with our own custom datasets. Some ideas we had were to create a virtual representation of ourselves using a LLM-based conversation agent, or to create a LLM-augmented knowledge base for our private notes and documents. While retrieval-augmented LLM are not new, there are very few existing implementations that are practical for our uses cases and are cost-effective. On a practical note, retrieval-augmented LLMs allow LLMs to generate responses based on custom domain-specific data without the need to perform parameter finetuning, which can often be expensive. This is particularly beneficial for cases where the data is private or is updated frequently. Overall, through this project we hope to learn more about the knowledge retrieval using LLMs, in-context question answering and prompt engineering.

Vector Database

A core part of our pipeline is the vector database (vectorDB), which we will introduce here briefly. In short, A vectorDB is a type of database that specializes in handling embeddings. Embeddings, in the context of language models, are mathematical representations of text where words or phrases with similar meanings are mapped to similar positions in a high-dimensional space. VectorDBs are simply databases provide optimized storage and querying capabilities for embeddings. There are many use cases of vectorDBs such as semantic search and long term memory for conversation agents. In our case, we use it for knowledge retrieval.

Architecture

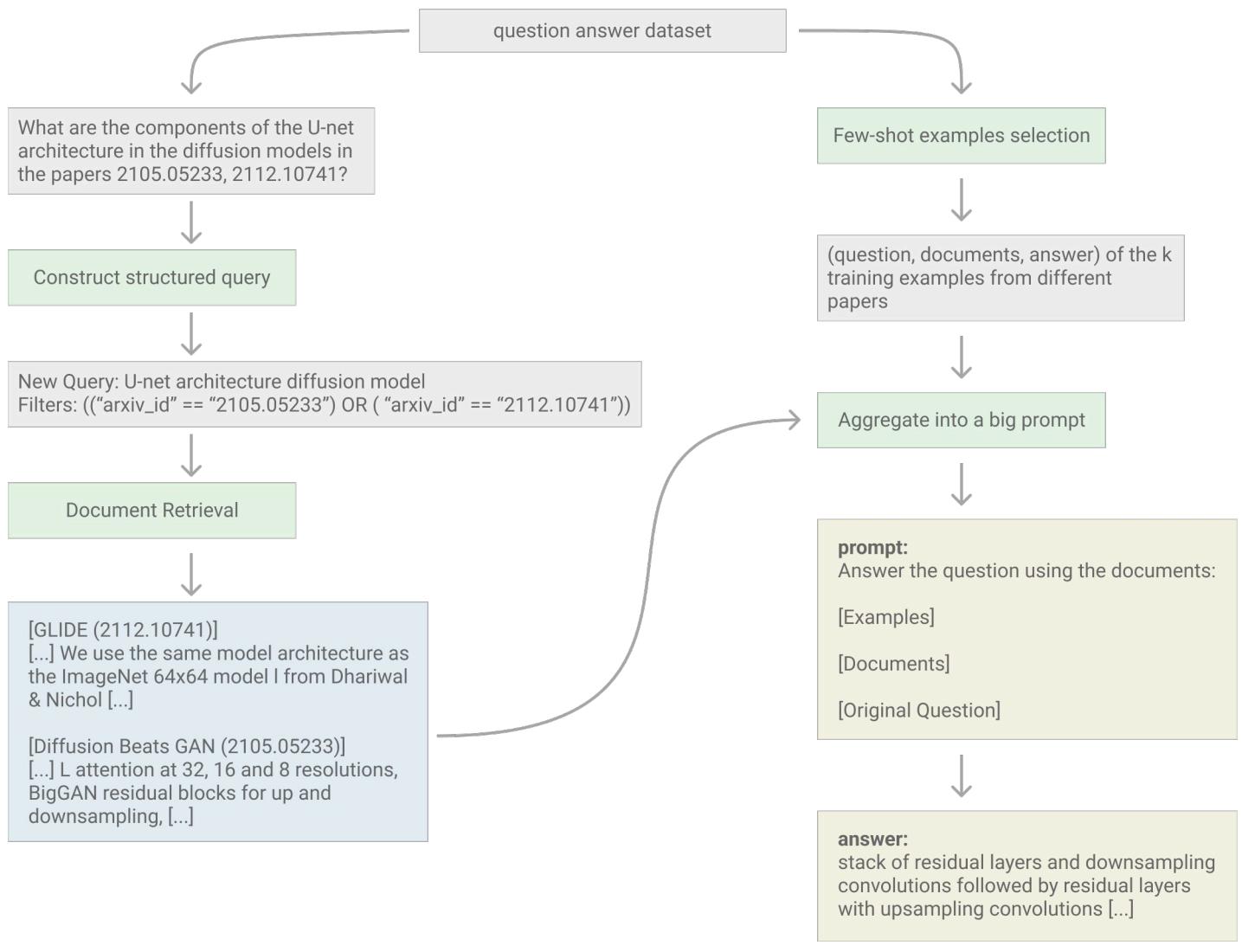
Our pipeline consists of three main components, a vectorDB, a LLM and the client interface. For the vectorDB, we use chromaDB, which is an open source vectorDB. The database is hosted on a serverless container using modal. For the embedding model, we are using instructor-xl, which is an instruction fine tuned embedding model based on a GTR model. The embedding model is also hosted on modal alongside the database. For the LLM, we use a proxy-based API of OpenAI's ChatGPT 3.5 turbo model. This API essentially allows us to use chatGPT for free with a daily rate limit. The rate limit is similar to using chatGPT directly through the browser, which is more than enough for our use case.

For ingestion, we first use document loaders from langchain to turn files stored on disk into

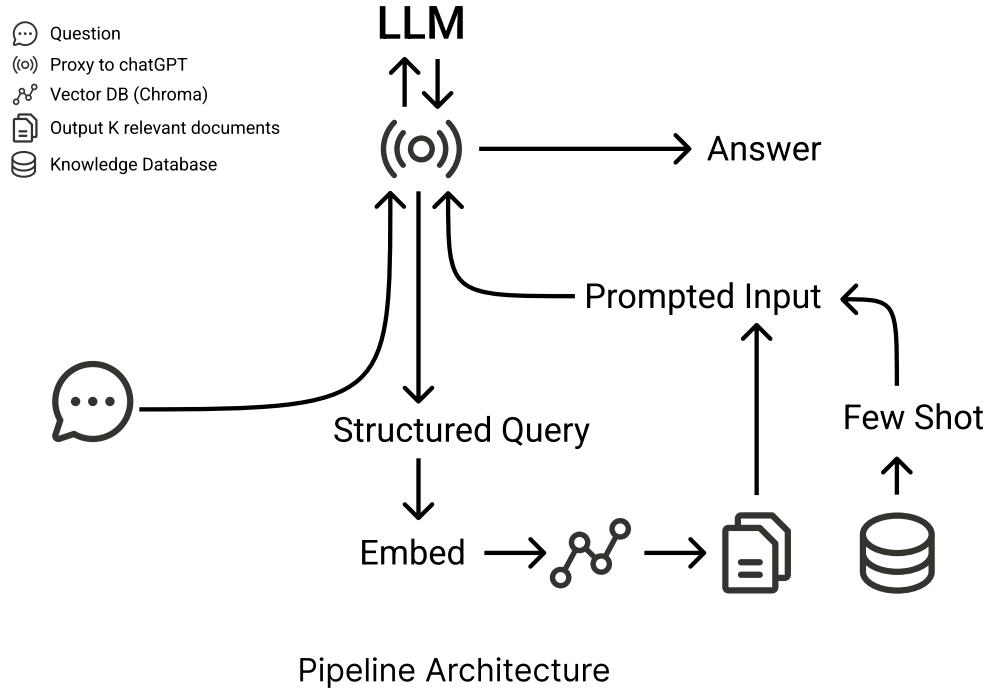
text. The text are then split into chunks of 1000-4000 characters (depending on the specific use case) with 200 characters of overlap. These chunks of text are then fed into the embedding model to generate embeddings. The embeddings are then stored in the vectorDB.

For querying, we first pass the query to the LLM to generate a more concise structured query that is more suitable for the vectorDB. The structured query is then passed to the embedding model to generate an embedding of the query. The embedding is then passed to the vectorDB to retrieve the top k most relevant documents. Our pipeline will also generate k-random examples from the training data to be used as few-shot examples. The documents are then combined with the few-shot examples and original query to form a large prompt. This prompt is then passed into the LLM for the final time to generate the final response.

Our data flow and pipeline architecture are illustrated in the diagrams below.



Dataflow



QASPER

Question	Type	%	Paper(s)	
What datasets do they use?	General	33.3%	1; 2; 3	
What other political events are included in the database?	Specific	66.7%	1706.01875	
Question	Answer	Type	%	Paper
What five dialogue attributes were analyzed?	Model; Confidence; Continuity; Query-relatedness; Repetitiveness; Specificity	Extractive	51.8%	1705.00571
Which neural architecture do they use as a base for their attention conflict mechanisms?	GRU-based encoder, interaction block, and classifier consisting of stacked fully-connected layers.	Abstractive	24.2%	1906.08593
Do they ensure the that the architecture is differentiable everywhere after adding the Hungarian layer?	Yes	Yes/No	13.9%	1712.02555
What language are the captions in?	N/A	Unanswer.	10.2%	1909.09070
Question	Evidence	Type	%	Paper
What new tasks do they use to show the transferring ability of the shared meta-knowledge?	To test the transferability of our learned Meta-LSTM, we also design an experiment, in which we take turns choosing 15 tasks to train our model with multi-task learning, then the learned Meta-LSTM are transferred to the remaining one task. The parameters of transferred Meta-LSTM, $\theta_m^{(s)}$ in Eq.(33), are fixed and cannot be updated on the new task.	Text	81.6%	1802.08969
How much does it minimally cost to fine-tune some model according to benchmarking framework?	Table 1	Table/Figure	11.6%	2002.05829
Do they recommend translating the premise and hypothesis together?	N/A	None	12.8%	2004.04721

QASPER Cont.

We use the QASPER dataset to evaluate the performance of our pipeline. QASPER is a dataset of 5,49 question over 1585 natural language processing (NLP) papers. Each question is written by an NLP practitioner who only reads the abstract and title of the paper. The questions are then answered by a separate set of NLP practitioners who have read the entire paper. The dataset consists of 4 types of answers: yes/no, extractive, abstractive, and unanswerable. This is illustrated in the table below. For more details about the dataset, check out the original paper.

Evaluation

For evaluation, we run three different versions of our pipeline over the test set of the QASPER dataset. The test set contains around 1500 questions over around 400 different papers. We chose the smaller-sized test set, as opposed to the training set since due to limitations in the rate limit of the API. We use the evaluation script provided by the original paper to evaluate our results. The three versions of our pipeline are:

LLM + 6 few-shot examples

LLM + 6 documents from vectorDB

LLM + 6 documents from vectorDB + 6 few-shot examples

We also attempted an experiment with an LLM without few-shot in context learning, however, that result was extremely poor and is not included here. We chose 6 documents and 6 few-shot examples as it gave us the best results out of the different combinations we tried, while keeping the context size within the 4000 token limit of chatGPT. For ingestion, we split the papers into chunks of 1000 characters with 200 characters of overlap. The evidence that is usually needed to answer the questions in the QASPER dataset are relatively short and do not require long chunks of continuous text. So, we picked a relative smaller chunk size. This also allowed us to reference more documents during the query phase, and increasing the probability of choosing the right document in order to answer the question. Our results are shown in the table below. Pipeline 1 is labeled as chatGPT. Pipeline 2 is labeled as our pipeline w/ zero-shot. Pipeline 3 is labeled as our pipeline w/ few-shot.

Model	Extractive	Abstractive	Boolean	Unanswerable	F1 Score
Human	58.9	39.7	79.0	69.4	60.9
LED-base	30.0	15.0	68.9	45.0	33.6
VisConde	52.3	21.7	86.2	48.3	49.1
ChatGPT	12.9	14.2	67.8	53.2	25.7
Our Pipeline w/ Zero Shot	37.5	20.0	10.2	48.0	30.8
Our Pipeline w/ Few Shots	50.3	24.4	71.8	72.2	49.2
SOTA					53.9

Results from our experiments compared to results from other papers.

Results

Based on the F1 similarity test reported by different papers and the test scripts we have ran on our pipeline and chatGPT, we have created the table below. The LED-base score is provided in the QASPER paper, while the VisConde score is provided in the VidConde paper. The SOTA score we are using in the table is the score of the CoLT5 model, which is currently the highest on the leaderboard as of June 1st 2023. Then we have ChatGPT's score with few shot examples, and the score of our pipeline with and without few shot examples as inputted prompt.

As we can see our pipeline out performs the VisConde model by just a little and have significant improvement against both the ChatGPT and LED-base score. Moreover, it is relatively close to the SOTA model, which is finetuned and trained with a lot of computation resources. Therefore, we are quite satisfied with our result.

Looking closer into each category, we can see that our pipeline performs similarly to the VidConde model at extracting information, which is both a little bit short of human testers. We can see that abstraction tasks are in general really hard for every model and our pipeline is still quite far from the human abstraction score. We also note that our pipeline underperforms in answering yes/no questions compared to both humans and the VisConde model, which means it is not nearly as good at understanding the material context as the other two. Lastly, we see that our pipeline performs significantly better than every other model at answering unanswerable questions, which means it hallucinates a lot less than those models. This is extremely important as a big part of the goal of our project is to make the LLM hallucinate as little as possible.

All in all, we are satisfied with the result our pipeline has achieved and we hope to better improve the abstractive ability of our pipeline down the line for further development. Another note we'd like to make here is that our pipeline with out few shots prompting performs significantly worse because it does not answer the question in the correct format. For example, it would answer a yes/no question like "yes, because ...", while the result should only have yes or no for better F1 similarity score. Therefore, adding the few shot example when prompting significantly improved the performance of our pipeline.

Limitations

Although we have achieved a decent performance, the project still has some limitations. Firstly, it cannot accommodate text inputs that exceed the context window size, which restricts the complexity and depth of the data we can process. Secondly, the system struggles to answer abstractive questions that necessitate reasoning with extensive context, limiting its problem-solving capability in certain scenarios. We can see that this is a common trend among every model and even is the worst category humans are performing in. Additionally, the employed similarity search method is imperfect and could overlook important context or provide irrelevant documents, impacting the accuracy and relevance of responses. Finally, we are using ChatGPT instead of our proprietary Large Language Model (LLM), which might limit our customization ability and overall control over the system. Specifically, we do not possess the ability to finetune the model unless we switch to an open source model and run it on our own server after fine tuning.

Next Steps

To build on our current project, we aim to run our own Large Language Model (LLM) on our own server to solve the rate limit and privacy issue we have right now using api calls. Moreover, we would like to use this pipeline in production with the project ideas we've mentioned in the motivation section, which would allow us to identify potential issues when this pipeline is deployed in different problem spaces. We would also like to tune our parameters such as the chunk sizes when processing the knowledge base to optimize for our performance (specifically the abstraction field). Lastly, we'd also like to use the 'chain of thoughts' method in prompting by integrating generated explanations from the examples used in our few-shot learning approach into the database. According to our research, this method has been proven effective in many papers when trying to improve the performance of a LLM.

Sources

1. langchain <https://python.langchain.com/en/latest/index.html>
2. A Dataset of Information-Seeking Questions and Answers Anchored in Research Papers <https://arxiv.org/abs/2105.03011>
3. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks <https://arxiv.org/abs/2005.11401>
4. CoLT5: Faster Long-Range Transformers with Conditional Computation <https://arxiv.org/pdf/2303.09752.pdf>
5. ChatGPT API FREE Reverse Proxy <https://github.com/PawanOsman/ChatGPT>
6. Chroma - the open-source embedding database <https://github.com/chroma-core/chroma>
7. Modal: End-to-end cloud compute <https://modal.com/>
8. instructor-xl <https://huggingface.co/hkunlp/instructor-xl>

Example Outputs

As a demo, we embedded around 10 papers related to the GLIDE paper and diffusion models. Here are some example outputs from our pipeline.

Question: What are the components of the u-net architecture used in diffusion models?

Answer: One component used in a U-Net architecture for diffusion models is self-attention layers, which have become important in recent literature. Another component is residual connections. The U-Net incorporates downsampling and upsampling to operate on larger spatial scales for increased computational efficiency, and features long-range skip-connections. The network is grouped into sequences of six layers, with most sequences separated by downsampling or upsampling.

Question: Why is classifier free guidance better than classifier-guided diffusion for the GLIDE model?

Answer: Classifier-free guidance is better than classifier-guided diffusion for the GLIDE model because it simplifies the model training pipeline and does not require training a separate classifier, which can reduce the need for noisy data. Additionally, classifier-free guidance allows a single model to leverage its own knowledge during guidance, which can improve sample quality. Finally, classifier-free guidance can avoid issues related to decreased diversity that can arise from using classifier-guided diffusion.

Question: What is adaptive group normalization?

Answer: Adaptive Group Normalization (AdaGN) is a layer used in deep learning architectures that incorporates the timestep and class embedding into each residual block after a group normalization operation, similar to adaptive instance normalization and FiLM. It is used to condition diffusion models on class labels and has been found to improve sample quality.

Question: What are the core results from the paper (2112.10741)?

Answer: The core results from the paper "GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models" include the exploration of text-conditional image synthesis using diffusion models and two different guidance strategies (CLIP guidance and classifier-free guidance), the demonstration of fine-tuning the model to perform image inpainting and text-driven image editing, and the comparison of their model's performance with other state-of-the-art text-conditional image generation models, showing superior results in terms of photorealism and caption similarity. Additionally, the paper addresses safety considerations and proposes filtering training images to reduce potential harm in disinformation or biased images.