DATA 586 Final Report

Tim Pulfer, Jacob Rosen, Dhun Sheth

Abstract

This report presents the development and evaluation of a Retrieval Augmented Generation (RAG) model, designed to enhance information retrieval and response accuracy for hockey-related queries. Traditional Large Language Models (LLMs) often struggle when queried about real-time information, creating a need for a more precise solution. Our RAG model integrates an external up-to-date, hockey-specific knowledge base with a generative LLM to improve data relevance and accuracy.

The methodology employs the RAG framework that combines a retriever mechanism with a generative model. The retriever uses vector embeddings of hockey data to fetch relevant information based on query similarity. The generative component, powered by the Gemma-7b LLM, converts the retrieved data into coherent responses. Experimental results show that the RAG model significantly outperforms standalone LLMs, especially in delivering up-to-date and accurate answers. The Gemma-7b model, in particular, demonstrated strong performance, effectively using the contextual data provided by the retriever.

The findings illustrate the effectiveness of the RAG model in specialized information retrieval scenarios and underscore the importance of coupling precise data retrieval with sophisticated language generation. Future work may focus on further refining the retrieval processes and fine-tuning the pre-trained LLM by training it again on more similar hockey data for better results. This project highlights the potential of RAG models in transforming data retrieval for domain-specific applications.

Table of Contents

Introduction

As hockey enthusiasts who frequently participate in daily fantasy leagues, NHL sports betting, and regular hockey-related debates, we often require a way to look up hockey-related data or questions. Although a general search engine, such as Google, can be used, it typically returns not just relevant hockey-related sources, but also countless speculative or tabloid-type information. This adds extra overhead cost required to filter the results and evaluate if the information is credible or not. Ideally, we would like a more efficient way where the returned results are more relevant and do not involve extra overhead cost.

Intuitively, a solution for the above problem could be an LLM. LLMs are good for generation tasks and are trained using large amounts of multimodal data. However, some critical shortcomings of LLMs include generating outdated information or being biased towards the data the model was trained on [1]. These shortcomings cannot be overlooked because having up-to-date information is critical for making fantasy league decisions such as which players to draft or drop. In addition, having the model generate biased answers is also problematic because then extra care must be taken while compiling the training data to ensure it's not biased.

One method which has grown in recent popularity for addressing these issues with stand-alone LLMs is a Retrieval Augmented Generation (RAG) model [2]. The model uses a retriever to search and return information from an external knowledge base (typically more specific and updated data compared to the initial training data) to supplement the knowledge of a generative LLM. The results are a more accurate and up-to-date response.

This allows for a more tailored approach to answering the previous questions that would otherwise be prompted in the search engine. By creating an up-to-date factual hockey-focused knowledge base, combined with an LLM for the generation step, we expect the responses from the RAG model to be more relevant and a faster way to get the same information versus using a search engine or an LLM on its own.

Methodology

Background

A retrieval augmented generation model (RAG) is a newly proposed framework that is used to enhance an existing large language model (LLM). What makes RAG models so groundbreaking is their use of an external knowledge base and how the model integrates this database with the existing LLM model [3]. This means a RAG model can be broken down into two key components, the retrieval model and the generative model. The retrieval model is responsible for interacting with the external knowledge base and gathering relevant information regarding the prompt. These retrieval models excel at obtaining relevant and accurate information but cannot generate or create unique content. Meanwhile, the generative model is an existing pre-trained LLM, these models are trained on large amounts of data allowing them to learn the structure of natural language. These generative models are excellent at creating coherent text but can tend to make up information that lacks factual accuracy [4].

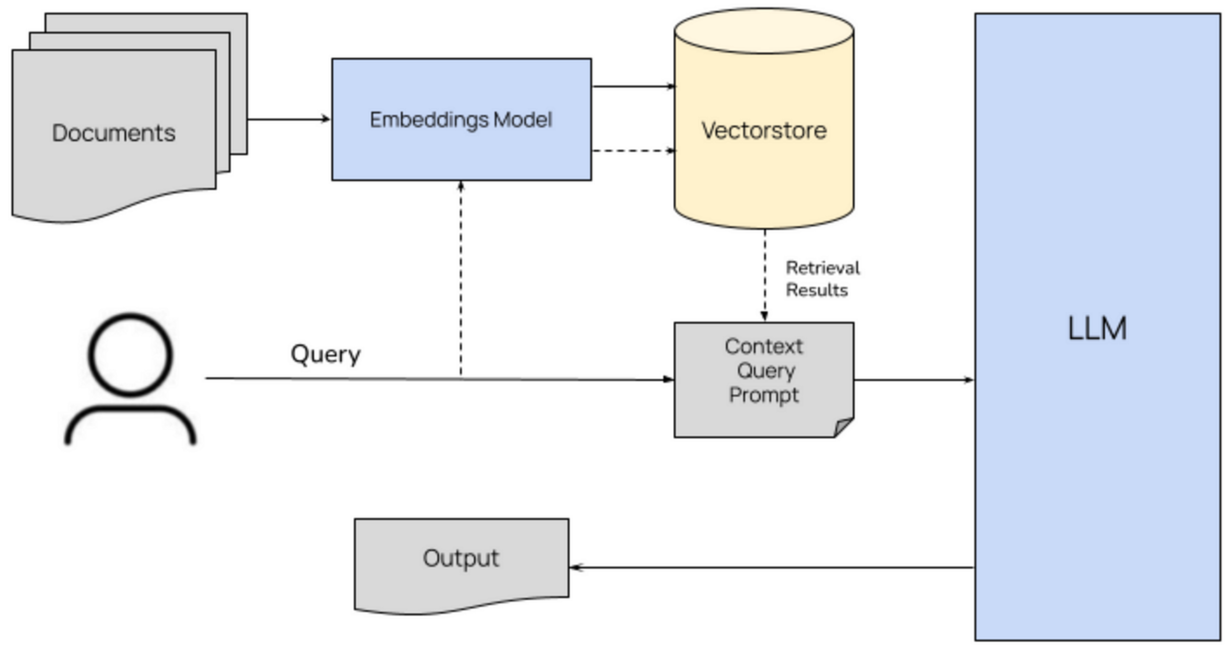

Figure 1: How RAG works. [5]

External Knowledge Base (how database is complied and partitioned)

Our external knowledge base consists of the top NHL player statistics about the 2023 NHL season. This information is stored as full sentences in a multipage .pdf file. This external knowledge base can easily be updated whenever needed, whether that be for a new NHL season or with completely different information to change the model's purpose. This ability to augment the model’s knowledge without retraining it makes RAG an essential component to any LLM that seeks to generate real, accurate, and up-to-date generations.

The pdf file is imported and each page is processed and chunked into sentences to make embedding easier. Each sentence within the document is then converted into a vector embedding, which is a numeric representation of the sentence in sparse vector space. Converting the sentences into embeddings allows the retriever to compute the similarity between the user query and the sentences of each document. The ‘all-mpnet-base-v2’ model, which is a sentence-transformers model on HuggingFace [6], was used to embed the sentences.

Retriever

The goal of the retriever is to search for relevant context related to the user query in the external knowledge base and return it to the generative LLM. The retriever uses dot product similarity to compare each embedding in the knowledge base against the embedding of the user query. In general, embeddings that are similar to each other in vector space should contain relatively similar linguistic information. Therefore, embeddings that rank highest in the similarity measure are most likely to contain relevant information to the query. The k-highest ranking embeddings are then passed as context to the generative model.

Generator

For our generator, we will be using ‘gemma-7b-it’ which is a state-of-the-art open-source generative language model developed by Google [7]. The model has over 8 billion parameters and was trained for a range of NLP tasks in English. Gemma-7b is a comprehensive yet lightweight model, making it a good fit for our task. The role of the generative model is to extract the relevant information passed to it from the retriever and attempt to answer the query as accurately as possible. The context from the retriever is passed to the generative model as part of the model prompt, along with the query. Prompt engineering is necessary for inputting the query and context together in a format that the model can interpret. Examples of good model outputs were also included in the prompt. The generator then returns an answer to the user's query.

The RAG implementation described above was based on the implementation from Daniel Bourke [8].

Results

Experiment 1: How does the LLM model perform without any context from the retriever

Result:

Query:

```
"How many goals did J.T. Miller score?”
```

Output:

```
I am unable to access real-time information, therefore I cannot provide
you with the number of goals J.T. Miller has scored.
```

Because the LLM was trained on a variety of data, that may be outdated now, it is unable to answer the query directly. We hypothesize including the retriever with more updated information, the model will be able to answer the query.

Experiment 2: Testing Gemma 2b Model

Result:

Query:

```
"How many goals did Connor McDavid score?”
"What team does Mitch Marner play for?"
"How many games did Mika Zibanejad play?"
```

Output:

```
Answer:
The context does not provide any information about how many goals Connor
McDavid
scored, so I cannot answer this question from the provided context.
```

```
Context items:
Connor McDavid (Edmonton Oilers) appeared in 80 games, scoring 63 goals
and accumulating 90 assists, leading to 153 points. His ice time averaged
22:57 minutes per game with a shooting percentage of 17.2%.

Answer:
The context does not provide any information about the team Mitch Marner
plays
for, so I cannot answer this query from the provided context.
Context items:
Mitch Marner (Toronto Maple Leafs) featured in 78 games, scoring 32 goals
and 67 assists for a total of 99 points. His average ice time was 21:33
minutes per game, with a shooting percentage of 14.6%.

Answer:
The context does not provide the number of games played by Mika Zibanejad,
so I cannot answer this question from the provided context.
Context items:
Mika Zibanejad (New York Rangers) appeared in 82 games, netting 41 goals
and 44 assists, amassing 85 points. His average ice time was 21:07 minutes
per game, and his shooting accuracy was 16.7%.
```

Observing the output above, we see the retriever can add the correct context based on the initial prompt, however, the model is not able to pick the information up and answer the question.

We hypothesize the above issue may be a result of performance limitation of the LLM and a better LLM may yield more promising results.

<u>Experiment 3:</u> Testing Gemma 7b Model

<u>Result:</u>

Query:

```
"How many goals did Connor McDavid score?"
"What team does Mitch Marner play for?"
"How many games did Mika Zibanejad play?"
"What's Dhun's golf handicap?",
"Who are the top Toronto Maple Leafs scorers?",
"Name 3 Toronto Maple Leafs players"
```

Output:

```
Answer:
Connor McDavid (Edmonton Oilers) scored 63 goals in 80 games.

Answer:
Mitch Marner plays for the Toronto Maple Leafs.

Answer:
Mika Zibanejad played in 82 games. The text states that he appeared in 82
games, so the answer is 82 games.

Answer:
The text does not provide any information about Dhun's golf handicap,
therefore I cannot answer this query.

Answer:
The text does not mention any other scorers than Dhun on the Toronto Maple
Leafs, therefore I cannot answer this query.
Context items:
['Nazem Kadri (Calgary Flames) featured in 80 games, scoring 28 goals and
56 assists for 84 points. He spent an average of 19:10 minutes on the ice
per game with a shooting percentage of 13.2%.', 'Jonathan Marchessault
(Vegas Golden Knights) played 77 games, scoring 32 goals and providing 48
assists for a total of 80 points. He averaged 19:34 minutes on the ice per
game with a shooting percentage of 14.3%.', "Ryan O'Reilly (Toronto Maple
Leafs) featured in 80 games, scoring 24 goals and 49 assists for 73
points. He spent an average of 20:45 minutes on the ice per game with a
shooting percentage of 10.8%......

Answer:
Sure, here's the answer to the user query:  The 3 Toronto Maple Leafs
players
named in the text are Dhun, Ryan O'Reilly, and William Nylander.
Context items:
["Ryan O'Reilly (Toronto Maple Leafs) featured in 80 games, scoring 24
goals and 49 assists for 73 points. He spent an average of 20:45 minutes
on the ice per game with a shooting percentage of 10.8%.", 'Nazem Kadri
(Calgary Flames) featured in 80 games, scoring 28 goals and 56 assists for
84 points. He spent an average of 19:10 minutes on the ice per game with a
shooting percentage of 13.2%......
```

Using the Gemma-7b LLM over the Gemma-2b LLM does indeed result in the model being able to pick up correct information from the added context to give more accurate answers to the initial prompt.

Discussion

The main disadvantage of current LLMs is their tendency to be overly creative, this can lead to factual incorrect outputs in some cases. RAG models allow for the same creative generations that LLMs can offer while also grounding the generations in proven facts from an external knowledge base. This means that the accuracy of an RAG model is determined by two key components, the LLM that is chosen for the generator and the data in the external knowledge base. The improvement of using an external knowledge base was tested in Experiment 1 by entering the prompt directly into the LLM, and from the results, it was not able to generate a sufficient answer because the model was trained on data that would currently be outdated. Comparing this to the output while using the knowledge base to give the LLM additional context helped generate a better answer to the initial prompt.

Since the generator is responsible for producing the output, we found that changing this could have drastic results. As seen in the results between experiments 2 and 3, when we were using gemma-2b as our generator, the RAG model was not able to interpret the context and could not produce a meaningful output. However, once we changed our generator to use a more powerful gemma-7b model, we got an output that aligned better with our expectations.

When considering future improvements to our RAG model there are a few areas that would likely yield the greatest results. Since we saw such dramatic results increase from changing our generator function to a more powerful LLM, it can be assumed that using an even better LLM would yield better results. It is widely agreed upon that OpenAI currently has one of the best LLMs, using this model as the generator function may result in even better results for our RAG model. In addition, fine-tuning the pre-trained LLM on data similar to the external knowledge base or expected prompts may help the model handle future prompts more effectively. Other areas of improvement include improving the retriever by using another calculation method for the similarity index to pull more relevant context or trying different prompt formats.

To conclude, the above RAG model is an improvement from the standard pre-trained Gemma-7b LLM and has reached a satisfactory stage where it can answer basic questions using the external knowledge base to access updated and more relevant information. Although there are areas for improvement, the advantages of using a Retrieval Augmented Generation model can be seen very clearly.

References

[1] MongoDB. (n.d.). Retrieval-augmented generation. Retrieved April 17, 2024, from https://www.mongodb.com/basics/retrieval-augmented-generation

[2] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2021, April 12). *Retrieval-augmented generation for knowledge-intensive NLP tasks*. arXiv.org. https://arxiv.org/abs/2005.11401

[3] Analytics Vidhya. (2023). Retrieval-Augmented Generation (RAG) in AI. Retrieved from https://www.analyticsvidhya.com/blog/2023/09/retrieval-augmented-generation-rag-in-ai/#:~:text=Retrieval%20Augmented%20Generation%20(RAG)%20is%20a%20groundbreaking%20framework%20that%20enhances,up%2Dto%2Ddate%20responses.

[4] Colabdoge. (n.d.). What is RAG (Retrieval-Augmented Generation)? Retrieved from https://colabdoge.medium.com/what-is-rag-retrieval-augmented-generation-b0afc5dd5e79

[5] Stackademic. (April 19, 2024). Rag and Vector Serach. Retrieved from https://blog.stackademic.com/mastering-retrieval-augmented-generation-rag-architecture-unleash-the-power-of-large-language-a1d2be5f348c

Sentence-transformers. (April 19, 2024). All-MPNet-base-v2. Hugging Face. Retrieved from https://huggingface.co/sentence-transformers/all-mpnet-base-v2.

Google. (April 21, 2024). Gemma-7B-IT. Hugging Face. Retrieved from https://huggingface.co/google/gemma-7b-it.

[8] Bourke, D. (n.d.). Simple Local Rag. GitHub. Retrieved April 21st, 2024, from https://github.com/mrdbourke/simple-local-rag