2024년 09월 19일

AUTHOR

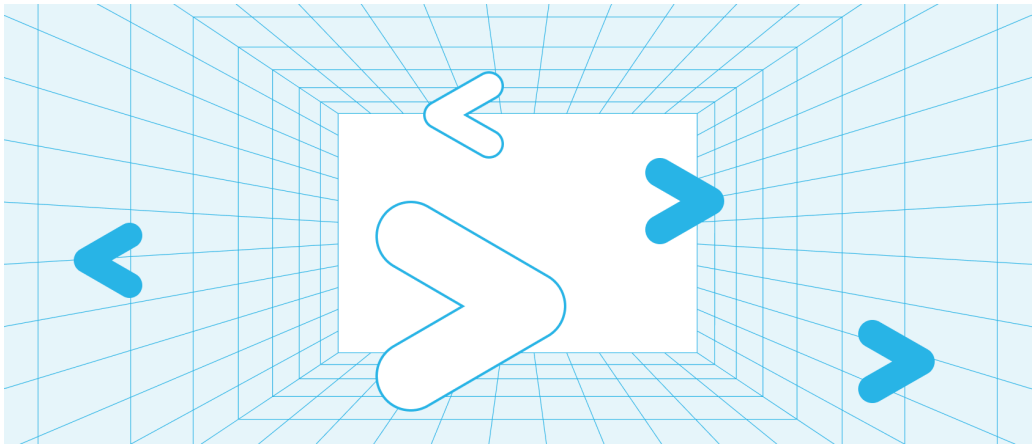
Daniel Campos

Gaurav Nuti
Danmei Xu

SHARE

Straight A's: You Can't Spell RAG Without Augmentation

Gen AI



Retrieval-augmented generation (RAG) is a significant advancement in AI, combining the strengths of information retrieval and generative models. In a typical setup, RAG systems first retrieve relevant documents or data from a large corpus and then generate a coherent and contextually accurate response by using the retrieved context. This approach enhances the accuracy of AI-generated responses by grounding the generation in the relevant context, making it valuable for applications like customer support, content creation, and research assistance.

While effective, deploying RAG outside of a demo can be fickle and require expertise to get it right. Having experienced the difficulty in running production-scale RAG workloads, Snowflake wants to provide its customers with the ability to chat with their data in a few clicks without any of the complex and cumbersome deployment pipelines. By taking care of the complexity behind the scenes, we are excited to empower our customers to focus on the benefits of applying RAG workflows, rather than the details of a pipeline.

When looked at from 30,000 feet, RAG workloads are simple, right?

There are two steps: (R) retrieve relevant documents and give them to a language model to generate (G), and it will be able to figure everything out.

A deployment could be achieved in a few minutes by simply calling a search API, then feeding the results to an inference/chat API and streaming the response back to the client. But wait — what happened to the A in RAG? Augmentation — what controls how you marry retrieved results with generation — is the glue. In our experimentation, we have found that RAG without carefully crafted augmentation can be fraught with errors and lead to frustration with users and developers.

AUTHOR

Daniel Campos

Gaurav Nuti

Danmei Xu

Getting RAG to work effectively is complex for several reasons. Firstly, ensuring high retrieval quality is crucial because the generative model's response is only as good as the retrieved information. This requires sophisticated search algorithms and fine-tuning for different data sets. Generated responses need citations to their sources, and citation accuracy is a major hurdle. The model must accurately reference its retrieved documents to support its answer. This cannot happen in a separate step because latency is another significant issue. Large generative models often have high response times, so waiting for the entire response to be generated before displaying it to the user is impractical. This necessitates streaming responses, which complicates post-processing and maintaining response coherence.

Not only do we need retrieval and generation to work well, but they need to work well together. This integration is critical to ensure the retrieved information seamlessly informs the generative model's responses.

tive combination, the output can become disjointed and lack coherence, leading to a poor user experience. Effective orchestration is essential but difficult, involving tasks such as context truncation, model selection, and prompt engineering. Balancing the right amount of context without overwhelming the model, choosing the most suitable generative model, and crafting prompts that minimize hallucinations are all challenges that must be addressed.

Despite these challenges, our team at Snowflake has developed innovative solutions to make RAG a seamless and powerful tool for our users. Read along to see the bag of tricks we employed to make RAG come together into a compelling experience.

Context is the door and good retrieval is the key

A good generation with the wrong context is very unlikely to meet user needs. What good is a good generation if the premise is wrong? The first step in making RAG work is getting the R (retrieval) part to work well. It is vital here because of the high sensitivity of the generative model on results being in the top position, as we will discuss later. Before we dove

into improving and understanding retrieval quality, we started with the most crucial step in any RAG pipeline: labeled data.

While labeling your data can seem daunting, creating a small iteration set can be quite fast. We brought together our search team and an existing corpus of documents (internal IT documents) with which we were familiar and asked people to write the queries that they would issue to reach a given document. This set of query document pairs is commonly called a QREL (Query Relevance label). You can pair your QRELs with metric tools like **ir_measures**, which builds on decades of retrieval research from **NIST's TREC** for quick and effective metric-driven development. We found that the retrieval set doesn't need to be huge, and in many cases, a few hours of labeled data can be highly sensitive to all the parameters in RAG. If getting people to label data proves to be cumbersome and you have an example set of queries, recent research suggests that you can use an **LLM as a judge** to create synthetic labels as a sufficiently sensitive proxy.

AUTHOR

Daniel Campos

Gaurav Nuti

Danmei Xu

Once we had our retrieval evaluation data set, we focused on optimizing our retrieval. We worked with **Cortex Search Service**, which is powered by Snowflake's Arctic Embed M, the best practical embedding model available, along with a traditional keyword index and a powerful reranker. The interplay between how these systems interact can be quite data set-dependent and we found that optimal retrieval parameters, such as keyword importance and retrieval depth, can see wide variability across sets. In practice, we found popular methodologies, like Reciprocal Rank Fusion, lead to subpar outcomes, as they did not support individual retrievers' high confidence in any document. Instead, we leverage a tuned set of per-source coefficients (discoverable via a library like **ranx**), which favors your data set's optimal performance. Aside from retrieval tuning, we also found high variation in optimal reranking depth using a cross-encoder. In some data sets, reranking anything more than your filter set (i.e., the documents given to the generator) introduces artificial noise, while other data sets improve the deeper you go. Such configuration and per-data set tuning will be soon available as a feature within Cortex Search. When evaluated on our sample IT data set we found it optimal to rerank the Top 8 results and per-source coefficients, which favored the embedding model slightly.

The special needs of RAG workloads are driven by limitations in generative models. While most claim large context lengths, up to 200k tokens, models have substantial position bias (**they favor top and bottom items instead of middle**), and the performance also degrades when you add more information to the context, as shown below. When we evaluated performance on our sample IT data set, we found that the

optimal number of retrieved chunks, which we refer to as the generation set, was just four. Increasing the size of the generation set with additional documents led the model to generate wrong outputs more often.

AUTHOR

Daniel Campos

Gaurav Nuti

Danmei Xu

Source: Fig. 9 from Lost in the Middle: How Language Models Use Long Contexts

Irrelevant document truncation is another complexity to consider when cultivating your retrieval set. While you may find that model performance is optimal with *N* documents, including irrelevant documents within that set is likely only to distract the generator.

One solution to removing distracting documents is to remove irrelevant results. How can you tell when results are irrelevant? Look at the relative distribution in retrieval scores and cross-encoder scores. If you have four documents and their cosine distance to the query is tightly coupled — say, within 0.02 of each other — then all these documents are likely relevant. However, if those four documents include one that is 0.35 further away than the rest, this document is likely irrelevant and can be *truncated* from the generation set. If you focus on cross-encoder scores you can use similar logic and prune, not by absolute score but relative outliers, in terms of irrelevance.

Finally, once you have optimized all your results you may notice that your generation set can have multiple chunks from the same document. While you can treat these as independent documents, this can lead to user confusion when reading responses. Users often look to citations to get

per-document generation origin despite the citations focusing on chunks of text.

The solution here can be simple. If chunks in the generation set are from the same document and close to each other you can combine them, and with some finessing of your prompt, let your model treat these related but separated chunks as a single document. It's worth noting that this final optimization is more of a stylistic choice and is more likely to have an impact if your generation set is quite large.

Conclusion

RAG is a powerful tool that can enhance the accuracy of AI-generated responses. However, deploying RAG requires expertise and careful augmentation. By understanding the challenges and complexities of RAG, you can unlock its full potential. With Snowflake, you can focus on how RAG can change your workflows, instead of getting bogged down in the details. So, go ahead and give RAG a try.

In the coming weeks, we will discuss subtleties in RAG workloads, such as quality benchmarks, document processing, citation generation, and broader generative modeling.

AUTHOR

Daniel Campos

Gaurav Nuti

Danmei Xu

SHARE

SHARE

RELATED CONTENT

2024년 08월 08일

Snowflake Cortex Search: High-Quality, Performant Search and Retrieval for Enterprise AI

2024년 06월 17일

Moving Beyond MTEB and BEIR:

2024년 07월 18일

Snowflake Arctic Embed M v1.5: Hitting

Search and retrieval systems have always been a critical backbone for knowledge management in enterprises. These systems cater to use cases ranging from site search, product catalog or feed search,...

Full Details

Snowflake AI Research Joins Forces with the University of Waterloo to Evolve RAG and Retrieval Benchmarks

To accurately answer business questions using LLMs, companies must augment models with their data. Retrieval...

More

the ROI Sweet Spot for Enterprise Retrieval

Today Snowflake released the world's most pragmatic text embedding model for English-language search: arctic-embed-m-v1.5. Our...

Find Out How

Gaurav Nuti
Danmei Xu

AUTHOR

Daniel Campos

Gaurav Nuti

Danmei Xu

SHARE