Retrieval-augmented generation

Retrieval augmented generation (**RAG**) is a type of information retrieval process. It modifies interactions with a large language model (LLM) so that the model responds to user queries with reference to a specified set of documents, using this information in preference to information drawn from its own vast, static training data. This allows LLMs to use domain-specific and/or updated information.[1] Use cases include providing chatbot access to internal company data, or giving factual information only from an authoritative source.[2]

Process

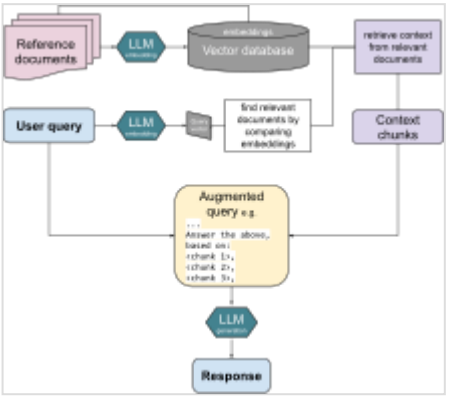
The RAG process is made up of four key stages. First, all the data must be prepared and indexed for use by the LLM. Thereafter, each query consists of a retrieval, augmentation and a generation phase.[1]

Indexing

The data to be referenced must first be converted into LLM embeddings, numerical representations in the form of large vectors. RAG can be used on unstructured (usually text), semi-structured, or structured data (for example knowledge graphs).[1] These embeddings are then stored in a vector database to allow for document retrieval.

Retrieval

Given a user query, a document retriever is first called to select the most relevant documents which will be used to augment the query.[3] This comparison can be done using a variety of methods, which depend in part on the type of indexing used.[1]



Overview of RAG process, combining external documents and user input into an LLM prompt to get tailored output

Augmentation

The model feeds this relevant retrieved information into the LLM via prompt engineering of the user's original query.[2] Newer implementations (as of 2023) can also incorporate specific augmentation modules with abilities such as expanding queries into multiple domains, and using memory and self-improvement to learn from previous retrievals.[1]

Generation

Finally, the LLM can generate output based on both the query and the retrieved documents.[4] Some models incorporate extra steps to improve output such as the re-ranking of retrieved information, context selection and fine tuning.[1]

Improvements

Improvements to the basic process above can be applied at different stages in the RAG flow.

Encoder

These methods center around the encoding of text as either dense or sparse vectors. Sparse vectors, used to encode the identity of a word, are typically dictionary length and contain almost all zeros. Dense vectors, used to encode meaning, are much smaller and contain far fewer zeros. Several enhancements can be made in the way similarities are calculated in the vector stores (databases).

- Performance can be improved with faster dot products, approximate nearest neighors, or centroid searches.[5]
- Accuracy can be improved with Late Interactions.[6]
- Hybrid vectors: dense vector representations can be combined with sparse one-hot vectors in order to use the faster sparse dot products rather than the slower dense ones.[7] Other methods can combine sparse methods (BM25, SPLADE) with dense ones like DRAGON.

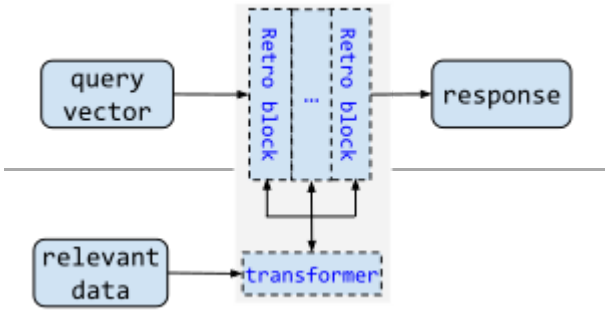
Retriever-centric methods

These methods focus on improving the quality of hits from the vector database:

- pre-train the retriever using the Inverse Cloze Task.[8]
- progressive data augmentation. The method of Dragon samples difficult negatives to train a dense vector retriever.[9]
- Under supervision, train the retriever for a given generator. Given a prompt and the desired answer, retrieve the top-k vectors, and feed those vectors into the generator to achieve a perplexity score for the correct answer. Then minimize the KL-divergence between the observed retrieved vectors probability and LM likelihoods to adjust the retriever.[10]

- use reranking to train the retriever.[11]

Language model



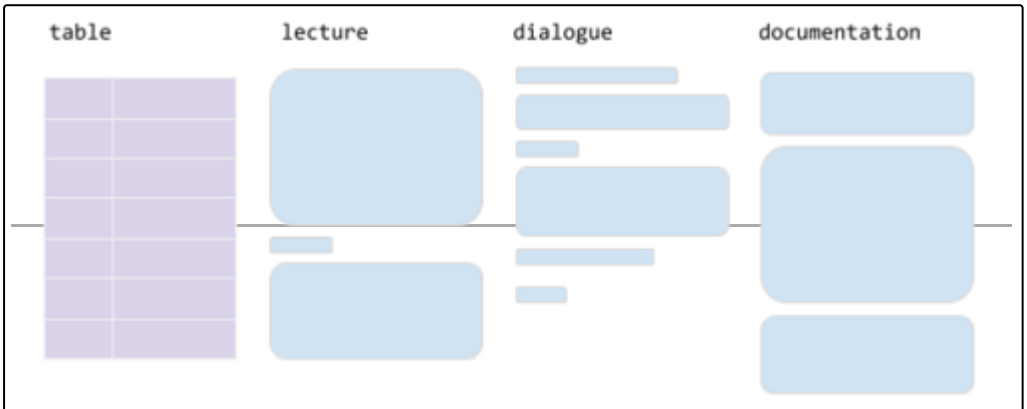
Retro language model for RAG. Each Retro block consist of Attention, Chunked Cross Attention, and Feed Forward layers. Black lettered boxes show data being changed, and blue lettering show the algorithm performing the changes.

By redesigning the language model with the retriever in mind, a 25-times smaller network can get comparable perplexity as its much larger counterparts.[12] Because it is trained from scratch, this method (Retro) incurs the heavy cost of training runs that the original RAG scheme avoided. The hypothesis is that by giving domain knowledge during training, Retro needs less focus on domain and can devote its smaller weight resources only on language semantics. The redesigned language model is shown here.

It has been reported that Retro is not reproducible , so modifications were made to make it so. The more reproducible version is called Retro++ and includes in-context RAG.[13]

Chunking

Converting domain data into vectors should be done thoughtfully. It is naive to convert an entire document into a single vector and expect the retriever to find details in that document in response to a query. There are various strategies on how to break up the data. This is called Chunking.



Different data styles have patterns that correct chunking can take advantage of.

Three types of chunking strategies are:

- Fixed length with overlap. This is fast and easy. Overlapping consecutive chunks help to

maintain semantic context across chunks.

- Syntax based chunks can break document up by sentences. Libraries such as spaCy or NLTK can also help.
- File format based chunking. Certain file types have natural chunks built in and it's best to respect them. For example, code files are best chunked and vectorized as whole functions or classes. HTML files should leave <table> or base64 encoded <img> elements intact. Similar considerations should be taken for pdf files. Libraries such as Unstructured or Langchain can assist with this method.

Challenges

If the external data source is large, retrieval can be slow. The use of RAG does not completely eliminate the general challenges faced by LLMs, including hallucination.[3]

References

1. Gao, Yunfan; Xiong, Yun; Gao, Xinyu; Jia, Kangxiang; Pan, Jinliu; Bi, Yuxi; Dai, Yi; Sun, Jiawei; Wang, Meng; Wang, Haofen (2023). "Retrieval-Augmented Generation for Large Language Models: A Survey". arXiv:2312.10997 (https://arxiv.org/abs/2312.10997) [cs.CL (https://arxiv.org/archive/cs.CL)].
2. "What is RAG? - Retrieval-Augmented Generation AI Explained - AWS" (https://aws.amazon.com/what-is/retrieval-augmented-generation/). *Amazon Web Services, Inc*. Retrieved 16 July 2024.
3. "Next-Gen Large Language Models: The Retrieval-Augmented Generation (RAG) Handbook" (https://www.freecodecamp.org/news/retrieval-augmented-generation-rag-handbook/). *freeCodeCamp.org*. 11 June 2024. Retrieved 16 July 2024.
4. Lewis, Patrick; Perez, Ethan; Piktus, Aleksandra; Petroni, Fabio; Karpukhin, Vladimir; Goyal, Naman; Küttler, Heinrich; Lewis, Mike; Yih, Wen-tau; Rocktäschel, Tim; Riedel, Sebastian; Kiela, Douwe (2020). "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks" (https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html). *Advances in Neural Information Processing Systems*. **33**. Curran Associates, Inc.: 9459–9474. arXiv:2005.11401 (https://arxiv.org/abs/2005.11401).
5. "faiss" (https://github.com/facebookresearch/faiss). *GitHub*.
6. Khattab, Omar; Zaharia, Matei (2020). " "ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT" " (https://dl.acm.org/doi/pdf/10.1145/3397271.3401075).
7. Formal, Thibault; Lassance, Carlos; Piwowarski, Benjamin; Clinchant, Stéphane (2021). " "SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval" " (https://www.semanticscholar.org/reader/6242b40d14746a418f30c4737d62d7649d08746f).
8. Lee, Kenton; Chang, Ming-Wei; Toutanova, Kristina (2019). " "Latent Retrieval for Weakly Supervised Open Domain Question Answering" " (https://aclanthology.org/P19-1612.pdf) (PDF).
9. Lin, Sheng-Chieh; Asai, Akari (2023). " "How to Train Your DRAGON: Diverse Augmentation Towards Generalizable Dense Retrieval" " (https://aclanthology.org/2023.findings-emnlp.423.pdf) (PDF).
10. Shi, Weijia; Min, Sewon (2023). " "REPLUG: Retrieval-Augmented Black-Box Language Models" " (https://aclanthology.org/2024.naacl-long.463).

11. Ram, Ori; Levine, Yoav; Dalmedigos, Itay; Muhlgay, Dor; Shashua, Amnon; Leyton-Brown, Kevin; Shoham, Yoav (2023). " "In-Context Retrieval-Augmented Language Models" " (https://aclanthology.org/2023.tacl-1.75).
12. Borgeaud, Sebastian; Mensch, Arthur (2021). " "Improving language models by retrieving from trillions of tokens" " (https://proceedings.mlr.press/v162/borgeaud22a/borgeaud22a.pdf) (PDF).
13. Wang, Boxin; Ping, Wei (2023). " "Shall We Pretrain Autoregressive Language Models with Retrieval? A Comprehensive Study" " (https://aclanthology.org/2023.emnlp-main.482.pdf) (PDF).

-