Week 5 Tuesday Paper 1: Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks

Explained by
What is Retrieval-Augmented Generation (RAG)?

Supplemental: vector databases and RAG - uses pinecone; gives simple code,
Vector Databases simply explained! (Embeddings & Indexes)

1. What are the LLM challenges the RAG paper addresses?
Ans: The challenges of LLMs that the RAG paper addresses include their limited ability to access and manipulate factual knowledge, difficulty in providing provenance for decisions, and challenges in updating their world knowledge.

2. Can RAG provide evidence for every answer it generates and how? Does this eliminate hallucination?
Ans: RAG models can provide evidence for each answer by retrieving relevant text documents as context for the generation, aiming to reduce hallucination. However, while this approach improves factual accuracy, it does not completely eliminate hallucinations.

3. What is the Retriever block? Explain the design, implementation, and role.
Ans: This architecture is composed of two main components:

Document Encoder (d(z)): This is a BERT-based model that generates dense representations of documents. The representation is a fixed-length vector embedding that encapsulates the information of each document in the database.

Query Encoder (q(x)): Similar to the document encoder, this is also a BERT-based model but is used to encode the query input into a dense vector representation. This enables the comparison between the query and document vectors.

The role of the Retriever is to compute the similarity between the encoded query and the encoded documents using a metric, which in the case of DPR is formalized through the exponential of the dot product. The process to find the top-k most relevant documents from the collection for a given query is framed as a Maximum Inner Product

Search(MIPS) problem. This problem can be solved approximately in sub-linear time, making the retrieval process efficient even over large document collections.

The DPR model is pre-trained on datasets such as TriviaQA questions and Natural Questions, where the goal is to find documents containing answers to given questions. This pre-training helps the Retriever understand which documents are likely to contain relevant information for a given query.

The document index, powered by these dense representations, serves as the non-parametric memory for the RAG system. It allows the model to dynamically access a vast amount of information outside its fixed parameters, significantly enhancing its knowledge base and retrieval effectiveness

4. What is the Generator block? Explain the design, implementation, and role.
Ans: The Generator block in the RAG framework is implemented using BART-large, a pre-trained seq2seq transformer model with 400 million parameters. This component can be modeled using any encoder-decoder framework, but BART is chosen for its effectiveness and efficiency in handling language generation tasks.
To combine the input “x” with the retrieved content “z” when generating from BART, we simply concatenate them.
BART has been pre-trained with a denoising objective, using a variety of noising functions to corrupt text in different ways and then learn to reconstruct the original text. This pre-training process enables BART to generate coherent, contextually relevant text based on the input it receives.
In the RAG model, the Generator block's role is to synthesize and generate the final output text based on the context provided by the retrieved documents and the initial query.
It functions as the "parametric memory" of the model, holding and applying the learned language patterns, grammar, and content to generate responses that are relevant to the combined input of query and retrieved content.

5. Explain the “Fast Decoding,” “Beam Decoder” and “Thorough Decoding”.
Ans: In the RAG framework, there are two distinct decoding methodologies used to generate the final text based on the retrieved documents and the input question. These are,

RAG-Token Decoding:

This approach treats the model as a standard autoregressive seq2seq generator.

For decoding, the modified probability is calculated for each token yi by considering the top-k documents retrieved and their relevance to the input.

This probability is then used within a standard beam decoder to generate the sequence of tokens for the answer.

The decoding process in RAG-Token considers the possibility of using different documents for each token in the output sequence, which can enhance the diversity and relevance of the generated text.

RAG-Sequence Decoding:

Unlike RAG-Token, RAG-Sequence does not break down the likelihood into per-token probabilities for conventional beam search.

Instead, it runs a beam search for each of the top retrieved documents independently, scoring each hypothesis based on the entire document's context and the input.

After obtaining a set of hypotheses, it performs an additional forward pass for each document that did not initially generate the hypothesis in its beam. This is to estimate the hypothesis's probability across all documents, a process referred to as "Thorough Decoding".

Due to computational demands, especially with longer sequences, a more efficient approach called "Fast Decoding" can be used. In this method, any hypothesis not generated in the beam search from a particular document is approximated to have near-zero probability, reducing the number of required forward passes.

Fast Decoding:

This is a simplified version of "Thorough Decoding" used in the RAG-Sequence approach.

In "Fast Decoding," if a hypothesis was not generated during the initial beam search for a particular document, it is assumed to have a negligible probability of being correct, thus avoiding additional computationally expensive forward passes.

This approximation significantly speeds up the decoding process while still maintaining a balance between efficiency and effectiveness in generating responses.

These decoding methods are integral to the RAG framework, as they directly influence the final output's relevance, diversity, and accuracy by leveraging the strengths of the retrieved documents and the generative capabilities of the seq2seq model

6. What different ablation studies were done? What is the conclusion from each? Explain intuitively why the conclusions obtained meet expectations.
Ans: Ablation studies explored the impact of different components like the retrieval mechanism and the number of retrieved documents. The studies showed that learned retrieval and the number of documents can significantly affect performance, confirming the importance of these components.

Retrieval Ablations: A key feature of RAG is learning to retrieve relevant information for the task. To assess the effectiveness of the retrieval mechanism, we run ablations where we freeze the retriever during training.
As shown in Table 6 in the paper, learned retrieval improves results for all tasks. We compare RAG's dense retriever to a word overlap-based BM25 retriever [53]. Here, we replace RAG's retriever with a fixed BM25 system, and use BM25 retrieval scores as logits when calculating p(z|x). For FEVER, BM25 performs best, perhaps since FEVER claims are heavily entity-centric and thus well-suited for word overlap-based retrieval. Differentiable retrieval improves results on all other tasks, especially for Open-Domain QA, where it is crucial.

Index hot-swapping: An advantage of non-parametric memory models like RAG is that knowledge can be easily updated at test time. Parametric-only models like T5 or BART need further training to update their behavior as the world changes.

RAG answers 70% correctly using the 2016 index for 2016 world leaders and 68% using the 2018 index for 2018 world leaders. Accuracy with mismatched indices is low (12% with the 2018 index and 2016 leaders, 4% with the 2016 index and 2018 leaders). This shows we can update RAG's world knowledge by simply replacing its non-parametric memory.

Effect of Retrieving more documents: Models are trained with either 5 or 10 retrieved latent documents, and we do not observe significant differences in performance between them. We have the flexibility to adjust the number of retrieved documents at test time, which can affect performance and runtime. Figure 3 in the paper shows that

retrieving more documents at test time monotonically improves Open-domain QA results for RAG-Sequence, but performance peaks for RAG-Token at 10 retrieved documents. It shows that retrieving more documents leads to higher Rouge-L for RAG-Token at the expense of Bleu-1, but the effect is less pronounced for RAG-Sequence.

7. What are vector databases? What are their advantages and applications?
Ans: Vector databases allow efficient storage and retrieval of high-dimensional vectors, supporting operations like similarity search, with applications in recommendation systems, search engines, and machine learning.

Vector databases are specialized storage systems designed for storing, indexing, and querying high-dimensional vectors, which are often used to represent data in machine learning models, particularly in the context of natural language processing, computer vision, and recommendation systems. These vectors are typically generated by embedding models, which convert complex data like text, images, etc. into numerical vectors that capture the semantic or contextual relationships between the data points.

Advantages of Vector Databases:

Efficient Similarity Search: Vector databases are optimized for high-speed similarity search, enabling them to quickly find the most similar vectors in the database to a given query vector. This is crucial for applications like semantic search, where the goal is to find items that are contextually similar rather than textually identical.

Scalability: They can efficiently handle large datasets and high-dimensional vectors, making them suitable for big data applications and services that require real-time performance.

Flexibility: Vector databases can accommodate vectors generated from various types of data, making them versatile tools for different domains and applications.

Space Optimization: Many vector databases use techniques like quantization and indexing to reduce the storage space required for high-dimensional vectors.

Applications of Vector Databases:

Recommendation Systems: They can improve the relevance of recommendations by finding items that are semantically similar to a user's past preferences.

Semantic Search Engines: Vector databases power semantic search capabilities, allowing users to find content that is contextually related to their search queries, not just textually matching.

Fraud Detection: In cybersecurity, vector databases can be used to identify unusual patterns or anomalies by comparing behavioral vectors.

8. What is pinecone and how is it used in RAG?
Ans: Pinecone is a cloud-based vector database that helps power AI for the world's best companies because it is fast and easy to use at any scale. It is used in retrieval augmented generation to gain non-parametric memory access to billions of documents. The Pinecone vector database lets you build RAG applications using vector search.

9. What questions do you have?

Ans: I understand the concepts very well but am a little hazy with the equations and the figure 1 for some reason. I will have to take a fresh look at it some other time.
The research paper is so good that I was able to understand most of it just by reading it.