Grounding LLMs for increased accuracy

How to use Generative AI to create Content

Grounding caters responses to specific information

- Grounding is the process of providing LLMs with a specific use case or data that is not available during the general part of the LLM's pretrained data
- Grounded LLMs are given a text base or examples and generate text based on those LLMs. This increases the accuracy of the responses and decreases the hallucinations

Why Grounding?

- Typically we think of LLMs as a general wealth of knowledge. They can do general reasoning and are good text engines. However they are not completely accurate
- They are also trained up to a certain time (September 2021 for GPT 3.5)
- By grounding we can use them more as text engines to extract relevant information. We can also give them sensitive data like corporate files, etc
- The normal process of grounding is called Retrieval Augmented Generation (RAG). RAG gets information relevant for a text and provides to the LLM as a prompt
- Fine tuning is another way to increase accuracy which creates a new models with task specific info, however this only allows a 1-2% increase in accuracy

Source:
https://techcommunity.microsoft.com/t5/fasttrack-for-azure/grounding-llms/ba-p/3843857#:~:text=Grounding%20is%20the%20process%20of,relevance%20of%20the%20generated%20output.

Common Use Cases

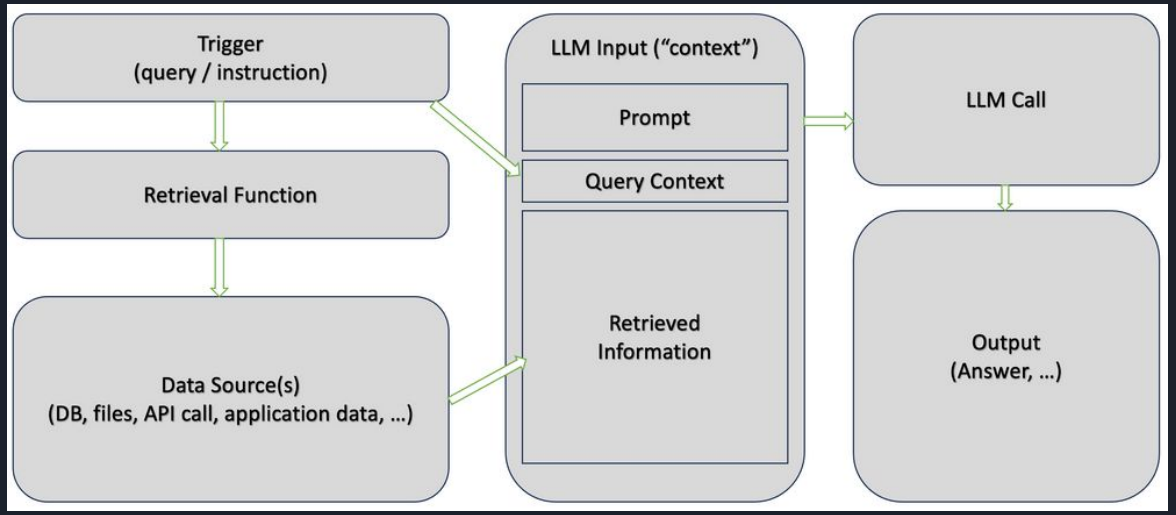
- Grounding can be used with web search and question and answer interfaces
- Google Generative Search Experience
- Microsoft Bing Chat
- Claude AI (Chat with PDF)
- Microsoft Copilot
- Google Duet AI

Simple RAG Model

The core LLM context is built with a prompt, query context and retrieved information.

When a user queries, it gets sent to a retrieval function that gets info from the data source. Then it builds all this info into an LLM context which is sent to the LLM

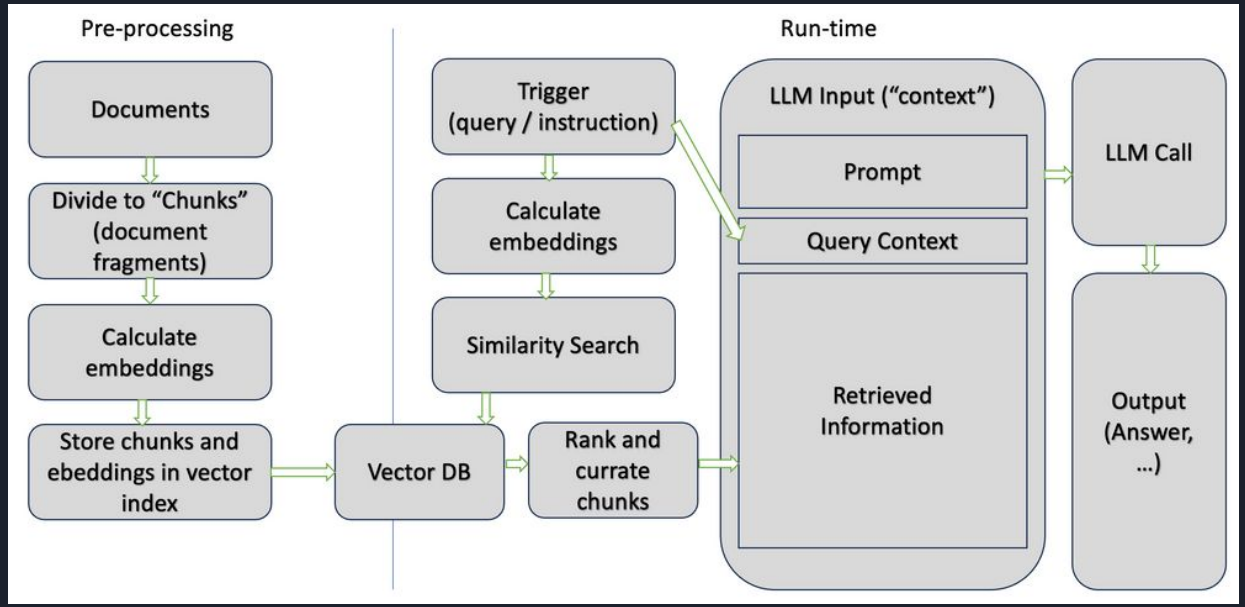
This is similar to a wrapper on a foundational model like Llama, Palm, or GPT



Simple Preprocessing Model

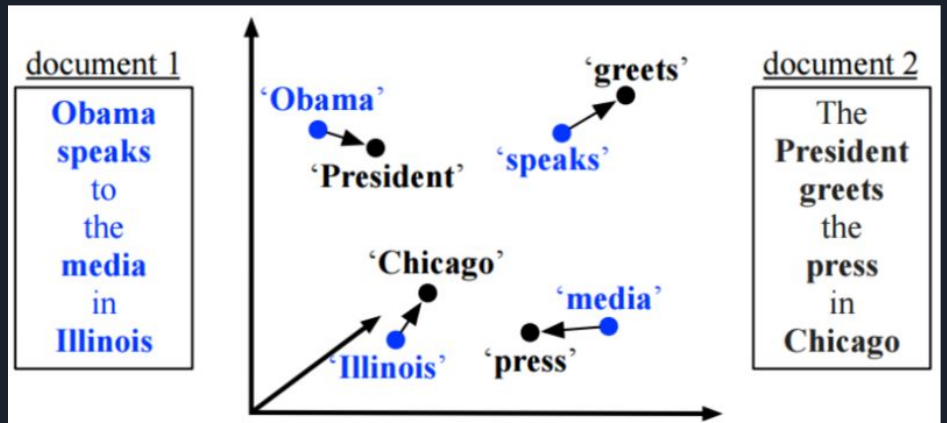
By preprocessing the documents into chunks and storing into a vector DB, we can make the RAG model run more efficiently.

This is a way of speeding up the RAG model execution time, once the items are preprocessed the remaining work is similar to the previous slide.



Embeddings find similarities between text

- Semantic search is a common way of finding relevant text
- For example the sentence Obama speaks to the media in Illinois is similar to The President greets the press in Chicago.
- This is because Obama is the same as President, speaks is the same as greets, media is the same as press, and Illinois is the same as Chicago.
- A technique like this makes it easier to find meaning in a large pile of text and in turn a popular preprocessing step in RAG models



Limitations

- Typically LLMs are a fixed context window size and models such as GPT 3, GPT 4, PaLM, and Llama might have too small of a context window to effectively perform RAG. One way around this is to rank information and only keep the most relevant info
- Ordering is important, sometimes incorrect sequences will cause wrong information. A general rule of thumb, is if it doesnt make sense to a human, it wont make sense to an LLM
- Formatting the LLM context can also change the output. We usually want to put spaces between each piece of relevant information

Tradeoffs

- The largest tradeoffs come in the form of speed vs cost vs accuracy
- For those building the tools there is a tradeoff of preprocessing vs runtime
- By introducing an extra step (LLM context building) to build the LLM query, this will cause a slowdown as well as use more compute resources
- In addition the extra processing will make tools more expensive, as a benefit you get greater accuracy from your queries.
- For developers there is a question whether we should preprocess the data or perform it at runtime, which is more expensive and slow.