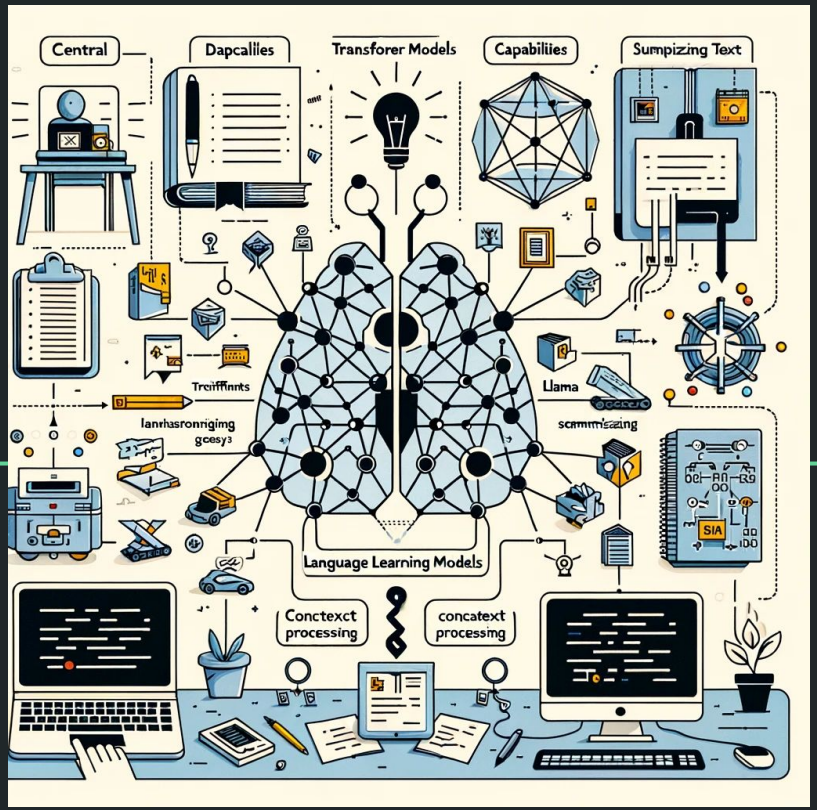


Enhancing LLMs with RAG

Alex Jacobs

Objective

- Overview of how LLMs (large language models) and RAG (retrieval augmented generation) work, and how they can work together
- Illustrate how an advanced AI application might work under the hood
- Short demo of an application focused on answer questions related to specific knowledge

Me

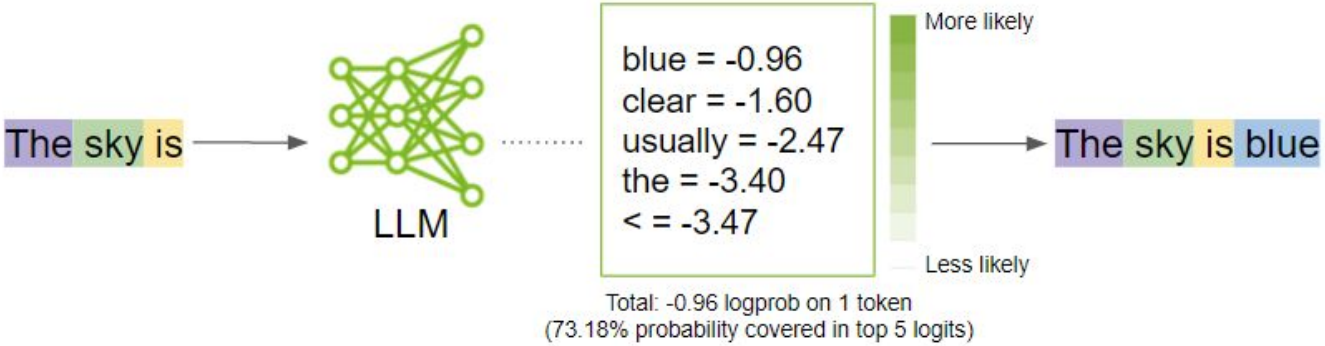
- Graduated from CofC 2016, B.S. in Data Science
- Backend engineer at Catalytic Data Science

Understanding LLMs (Large Language Models)

- LLMs are statistical models designed to understand, interpret, and generate human language by learning from vast amounts of text data.
- Based on transformer models which have a self-attention mechanism that allows for contextual understanding
- Capable of writing essays, summarizing text, generating code, and even creating powerpoint presentations
- Examples: ChatGPT (gpt3/4, LLama, Poe, Gronk)

Understanding LLMs (Large Language Models)

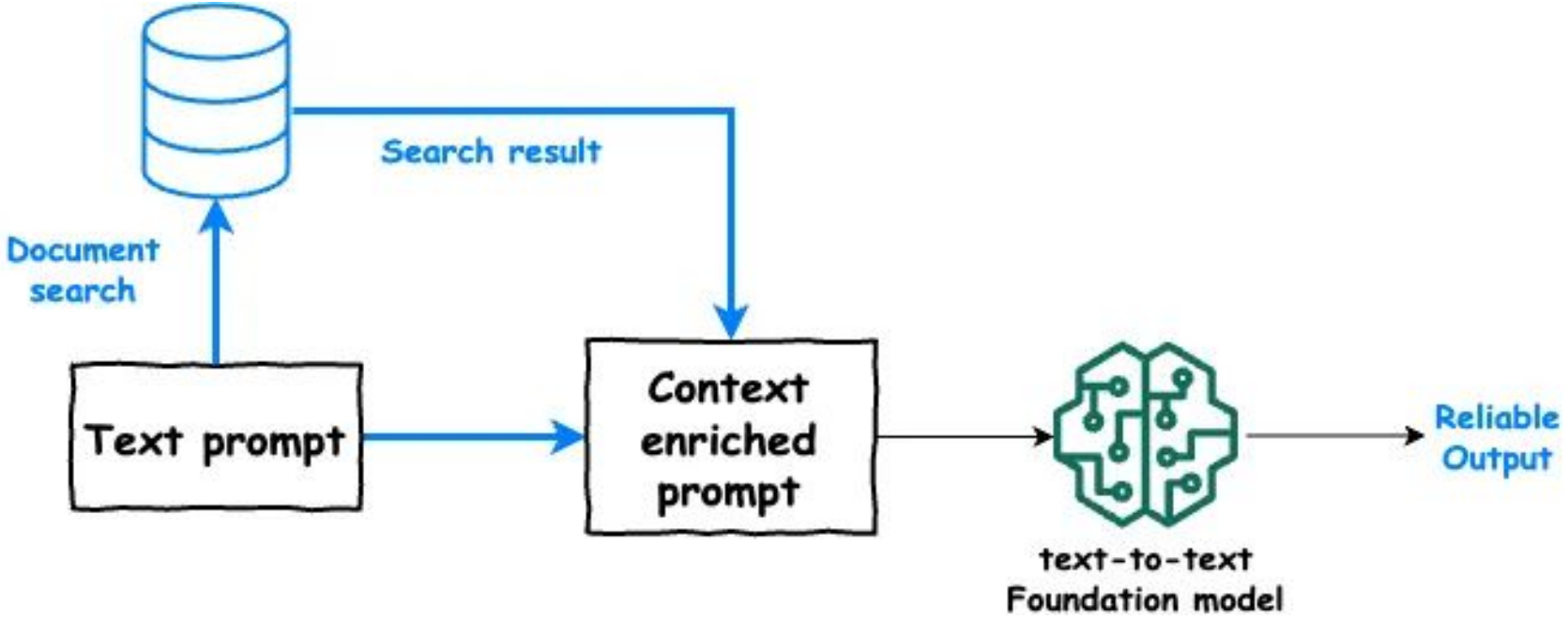
- Tokenization: process of breaking down text into smaller units (tokens) for the model to process. Each token represents a word or part of a word.
- Self-Attention Mechanism: allows the model to focus on different parts of the input sentence when predicting each word, understanding the context better.
- Training Proces: trained on large datasets, where they learn to predict the next word. This **unsupervised learning** helps them understand language structure and context.



RAG (Retrieval-Augmented Generation)

- RAG combines a retrieval system (to find relevant information) with a generative system (to create responses).
  - like having a librarian (retriever) and a storyteller (generator) working together.
- Provides additional, specific context to LLMs, enhancing their ability to generate relevant and accurate responses.
- Can improve the precision of answers, especially for questions requiring specialized knowledge.

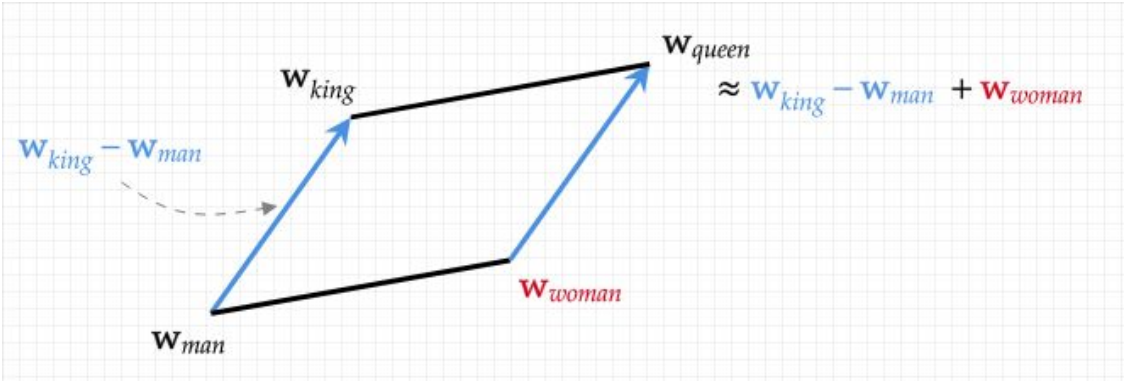
RAG (Retrieval-Augmented Generation)



Understanding Embeddings

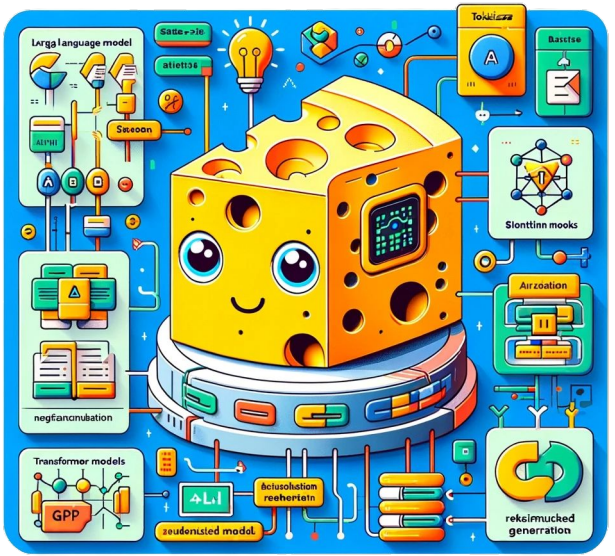
- embeddings (word vectors) are numerical representations of words or tokens, transforming them into vectors in a high-dimensional space.
- these vectors capture semantic and syntactic information about the words
- similar or related words have closer vectors in the embedding space, showing relationships like synonyms, antonyms, or thematic similarity.
- embeddings can capture complex relationships, like analogies (e.g., "man is to king as woman is to queen").

```
array([-0.5968882 , -0.33086956, -0.32643065, -0.3670732 ,  0.628059  ,
       -0.3692328 , -0.37902787, -0.12308089, -0.38124698, -0.03940517,
        0.2260839 ,  0.10852845, -0.2873811 , -0.42781743,  0.06604357,
       -0.07114276, -0.29775023, -0.99628943, -0.54497653, -0.11718027,
       -0.15935768,  0.09587188, -0.2503798 ,  0.06768776,  0.3311586 ,
        0.43098116,  0.06936899,  0.24311952,  0.14515282,  0.19245838,
        0.10462623, -0.45676082,  0.5662387 ,  0.69908774,  0.48064467,
        0.27378514, -0.45430255,  0.17282294, -0.40275463, -0.38083532,
        0.47487524,  0.31950948, -0.1109335 ,  0.2165357 ,  0.034114  ,
        0.05689918,  0.20939653,  0.15209009, -0.24204595,  0.03478364,
        0.1616051 , -0.5827333 , -0.47017908,  0.26226178, -0.11884775,
        0.40180743, -0.5173988 , -0.19270805,  0.660391  , -0.24518126,
       -0.42860952, -0.22274768,  0.4887834 ,  0.49302152,  0.38799986,
       -0.041193  , -0.38600504, -0.37632987,  0.04570564,  0.50462466,
       -0.14396502,  0.33490512, -0.15964787, -0.21363072, -0.25445372,
        0.52389127,  0.5747422 , -0.25075617, -0.5339069 ,  0.2582965 ,
       -0.16139959,  0.09748188,  0.04540966, -0.27768216, -0.51260656,
       -0.06189002, -0.54032195, -0.21863565,  0.06233869,  0.13287479,
        0.49741864,  0.1772418 ,  0.02064824, -0.04775626, -0.16804916,
        0.4643644 ,  0.5546319 ,  0.68051434,  0.7790246 ,  0.5617202 ],
      dtype=float32)
```



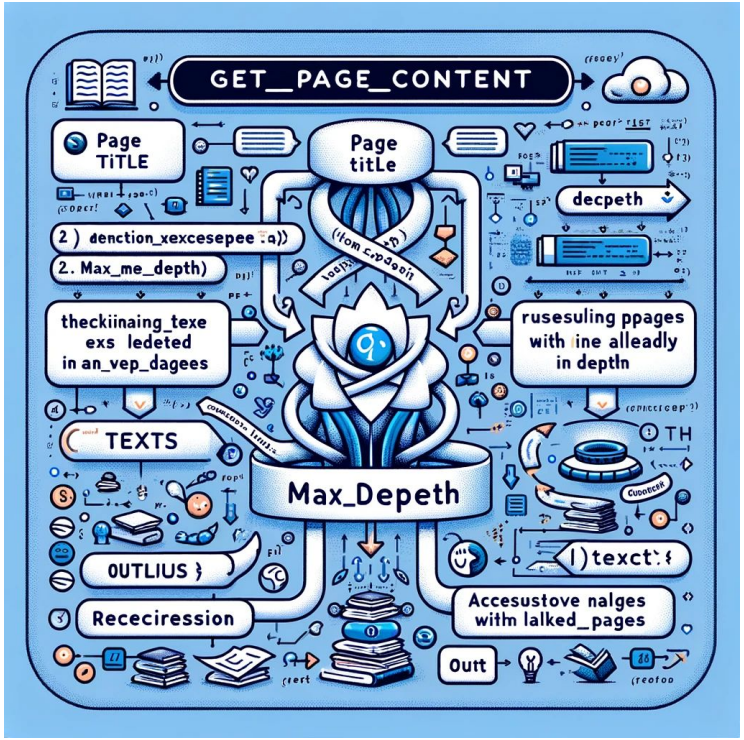
CheeseGPT

- Specialized chatbot that uses RAG and ChatGPT to provide expert-level information on cheese.
- Purpose is to deliver accurate and detailed answers to cheese-related queries, beyond what a standard ChatGPT could achieve
- Uses wikipedia to surface specific information on cheese



Data Collection: Scraping Cheese Wikipedia

- Starting from the “List of Cheeses” wikipedia page, grabbed every link recursively to a depth of 2 (every link on every page, and then every link on those pages too)
- ~50k pages.
- ~ 300k ‘documents’



Generate Embeddings

- Use langchain and openAi API to generate embeddings for all of our scaped wikipedia data
- Store in Redis (vector database)

```
(venv) alexjacobs@Alexs-MacBook-Pro ~/r/rag-experiments> rvl stats -i cheese

Statistics:
```

Stat Key	Value
num_docs	259448
num_terms	1.1367e+06
max_doc_id	259448
num_records	3.01998e+07
percent_indexed	1
hash_indexing_failures	0
number_of_uses	4
bytes_per_record_avg	6.93134
doc_table_size_mb	27.4646
inverted_sz_mb	199.628
key_table_size_mb	9.69387
offset_bits_per_record_avg	9.99252
offset_vectors_sz_mb	49.1662
offsets_per_term_avg	1.36671
records_per_doc_avg	116.4
sortable_values_size_mb	0
total_indexing_time	40837.8
total_inverted_index_blocks	1.38287e+06
vector_index_sz_mb	1536.03

Implement simple RAG

- Embed the user’s question, do a cosine similarity search to retrieve the documents with the most similar embeddings
- Include those documents in the context of the question passed to GPT4, and ask it to cite its sources

Demo

Questions?