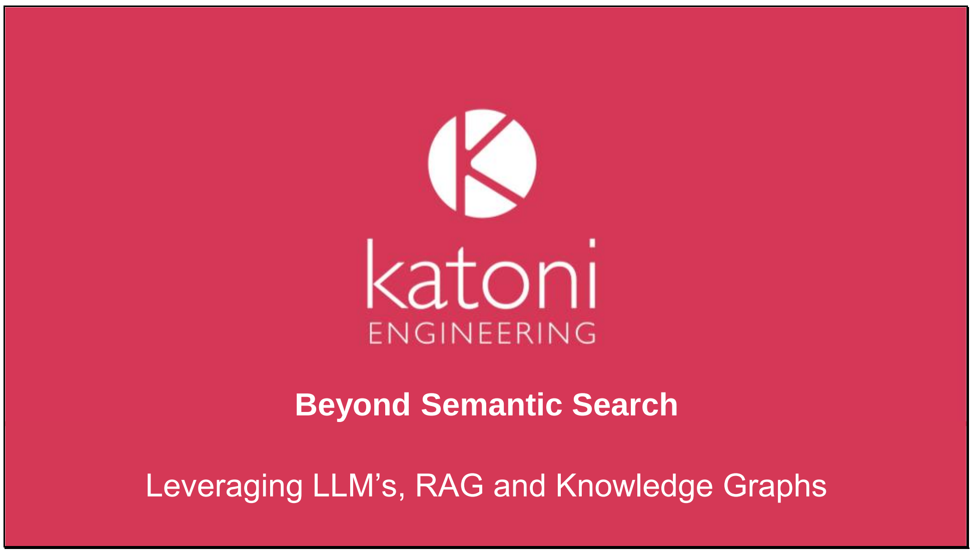


katoni ENGINEERING

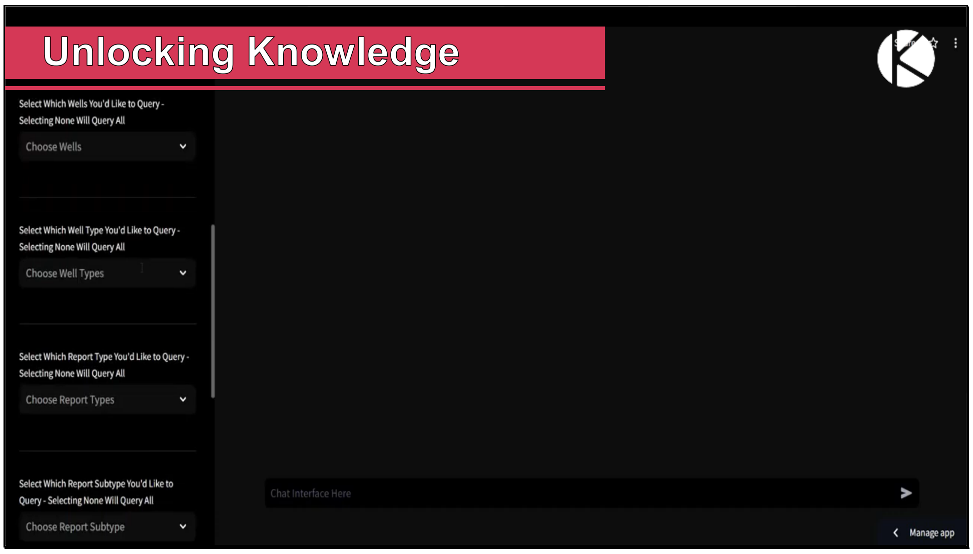
Beyond Semantic Search

Leveraging LLM’s, RAG and Knowledge Graphs

Why We Are Top of the Food Chain

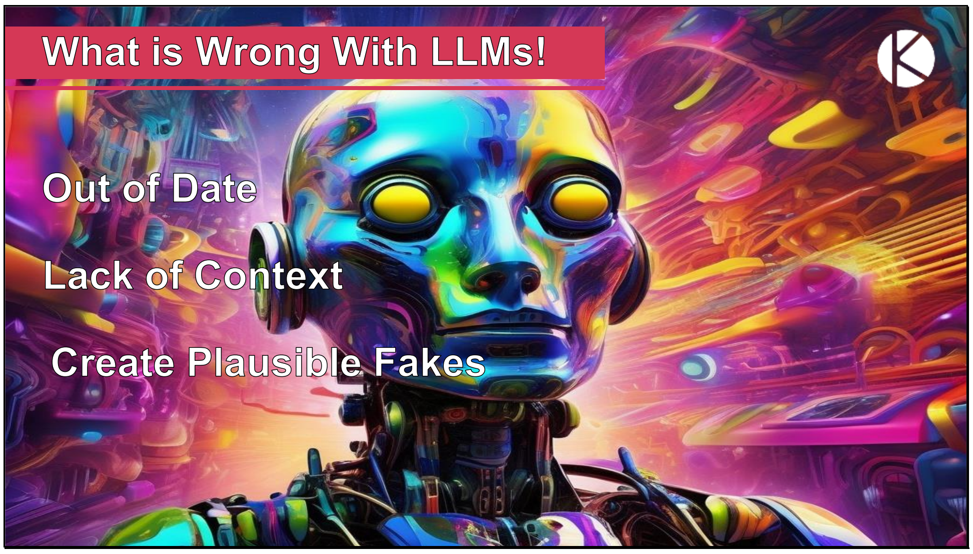
Our success as a species lies in something our ancestors did round the campfire, something we are doing right here right now: Storytelling. We've always been adept at sharing complex ideas, and we do it through language.

Consider the countless reports, emails, and memos you've written. Each one is a treasure trove of analysis and expertise. Imagine scaling this knowledge across an entire company, or even an industry—decades of collective knowledge, experience and wisdom. It's a vast resource, but accessing it effectively remains a challenge, even with tools like keyword or fuzzy search.


Unlocking Knowledge
Select Which Wells You'd Like to Query - Selecting None Will Query All
Choose Wells
Select Which Well Type You'd Like to Query - Selecting None Will Query All
Choose Well Types
Select Which Report Type You'd Like to Query - Selecting None Will Query All
Choose Report Types
Select Which Report Subtype You'd Like to Query - Selecting None Will Query All
Choose Report Subtype
Chat Interface Here
Manage app

As someone who is passionate about language, is endlessly curious, and with a background in exploration drilling, I was drawn to exploring the potential for language models in the energy industry. When ChatGPT gained popularity, I embarked on a personal research project using Equinor's open sourced Volve dataset. My goal was to create an information retrieval system that could extract valuable insights from this precious repository of drilling data.

Video Link: https://youtu.be/UO1tQ4SfgKQ?si=HfxCTb5C8iWM95cq



What is Wrong With LLMs!

Out of Date

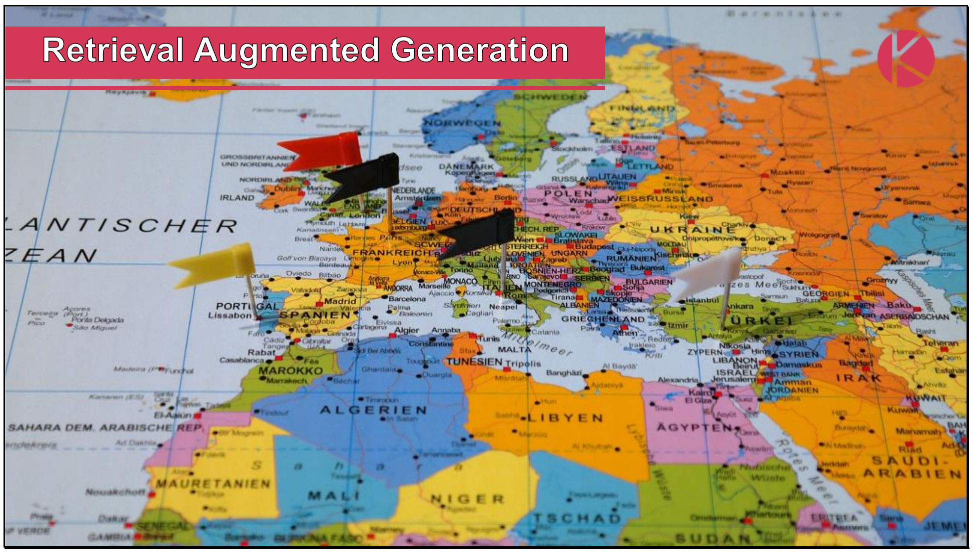
Lack of Context

Create Plausible Fakes

Human behavior and communication is intriguing isn’t it. That's why I’m drawn to natural language processing (NLP. When ChatGPT was released, I was initially very excited, but I feel we largely use it incorrectly.

Large Language Models aren't search engines. They're language calculators that predict the most likely next word based on patterns in their training data. This approach has limitations. For instance, the training data becomes outdated as soon as training is complete, LLMs often lack contextual understanding, Additionally, they can generate highly convincing but inaccurate information, you could say LLM’s are like some politicians you might be familiar with.

Instead of treating them as a reliable information repository, I believe we should view them as an interface to a recommendation engine.



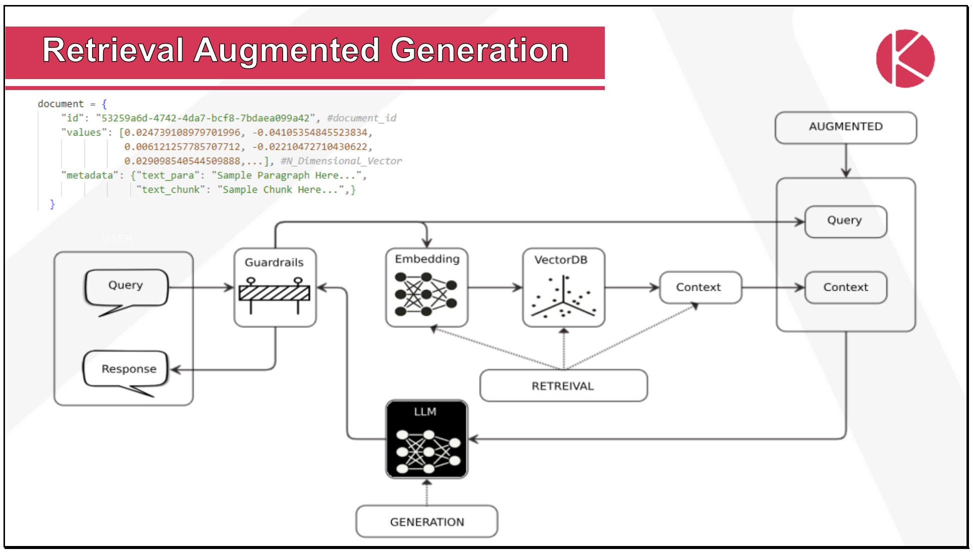
Making sure the system could access live relevant data, was critical. This would help minimize the risk of generating inaccurate or misleading responses. Using the LLM to construct database queries was too error prone, time to explore a more robust approach: Retrieval Augmented Generation (RAG).

This method leverages semantic search to retrieve relevant information from a knowledge base, enhancing the LLM's ability to generate informative and contextually appropriate responses.

How many people are familiar with RAG?

Transformer models, like those used in RAG, have two main components: an encoder and a decoder. The encoder part captures context within text by converting it into a list of numbers, hundreds even thousands of places long, depending on the model. These number lists, known as embeddings or vectors, can then be stored in a Vector Database.

RAG works by comparing these vectors to find the most similar ones. Think of it like a coordinates on a map. We could say that Aberdeen and Dundee are more similar to one another than they are to Riyadh by dint of geographical proximity. RAG does something similar, but instead of plotting places on a 2D map, it plots them in a much higher-dimensional space.



```
document = {
    "id": "53259a6d-4742-4da7-bcf8-7bdaea099a42", #document_id
    "values": [0.024739108979701996, -0.04105354845523834,
               0.006121257785707712, -0.02210472710430622,
               0.029098540544509888,...], #N_Dimensional_Vector
    "metadata": {"text_para": "Sample Paragraph Here...",
                 "text_chunk": "Sample Chunk Here...",}
  }
```

A basic RAG process looks something like this. There is part of this diagram that I haven't yet mentioned, Guardrails. Guardrails are the part of the system that ensures an LLM-based tool behaves predictably and reliably, a safety net if you will. You can design something yourself or use an off the shelf Framework like NeMo or Guardrails.ai.

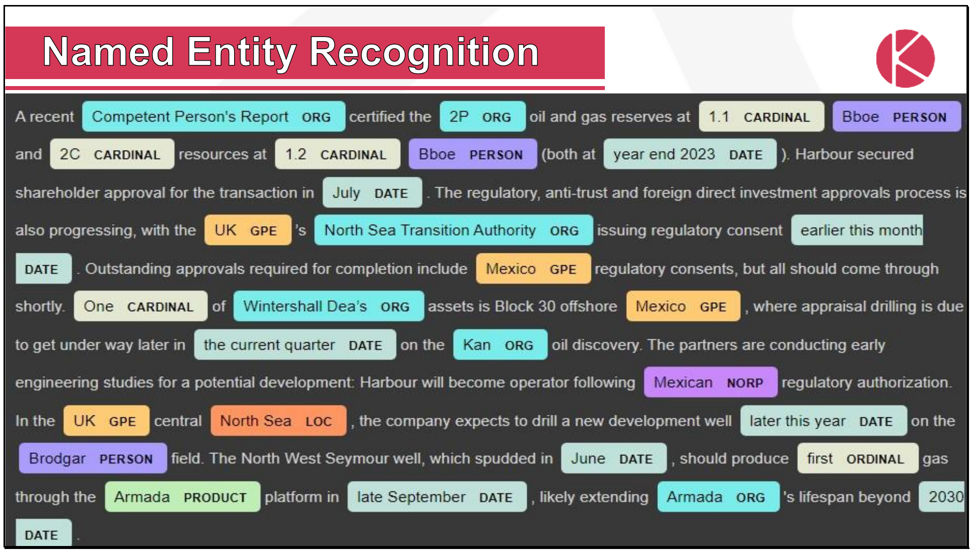
Back to our RAG process, A user passes a query to the system. The query gets screened by the guardrails to ensure it's appropriate and within the system's capabilities. The query is converted into vector and retrieves data from the vector database. The original query is then augmented with the most relevant information returned from the Vector Database, giving the LLM greater context, allowing it to generate a more accurate response. The response then goes back through the guardrails to ensure it's safe and on point.

Something we need to consider prior to adding data to the vector database, is how we split the documents, a chunking strategy. When splitting our documents into chunks, we want to capture the meaning clearly. Opting for sentence-level chunks might miss context, while large paragraphs can introduce noise or dilute the importance of specific words or phrases.

To address this, I used a technique called context-aware chunking. It leverages a document's linguistic structure to create meaningful and relevant chunks for comparison. These vectorized chunks are used for targeted search, but the full paragraph is returned for greater context when the LLM is generating a response.

What have we achieved? A powerful semantic search and summarisation system.

<u>Suggested Tools:</u>
ChromaDB
Pgvector
Pinecone
Weviate
LangChain
NLTK
Spacy



Named Entity Recognition

A recent Competent Person's Report ORG certified the 2P ORG oil and gas reserves at 1.1 CARDINAL Bboe PERSON and 2C CARDINAL resources at 1.2 CARDINAL Bboe PERSON (both at year end 2023 DATE). Harbour secured shareholder approval for the transaction in July DATE . The regulatory, anti-trust and foreign direct investment approvals process is also progressing, with the UK GPE 's North Sea Transition Authority ORG issuing regulatory consent earlier this month DATE . Outstanding approvals required for completion include Mexico GPE regulatory consents, but all should come through shortly. One CARDINAL of Wintershall Dea's ORG assets is Block 30 offshore Mexico GPE , where appraisal drilling is due to get under way later in the current quarter DATE on the Kan ORG oil discovery. The partners are conducting early engineering studies for a potential development: Harbour will become operator following Mexican NORP regulatory authorization. In the UK GPE central North Sea LOC , the company expects to drill a new development well later this year DATE on the Brodgar PERSON field. The North West Seymour well, which spudded in June DATE , should produce first ORDINAL gas through the Armada PRODUCT platform in late September DATE , likely extending Armada ORG 's lifespan beyond 2030 DATE .

While this basic RAG system is a valuable tool, it's important to acknowledge its limitations. One significant issue is that AI systems inevitably lose information from the original text. This loss can manifest in various ways, such as the omission of tangential details or the diminished significance of specific keywords.

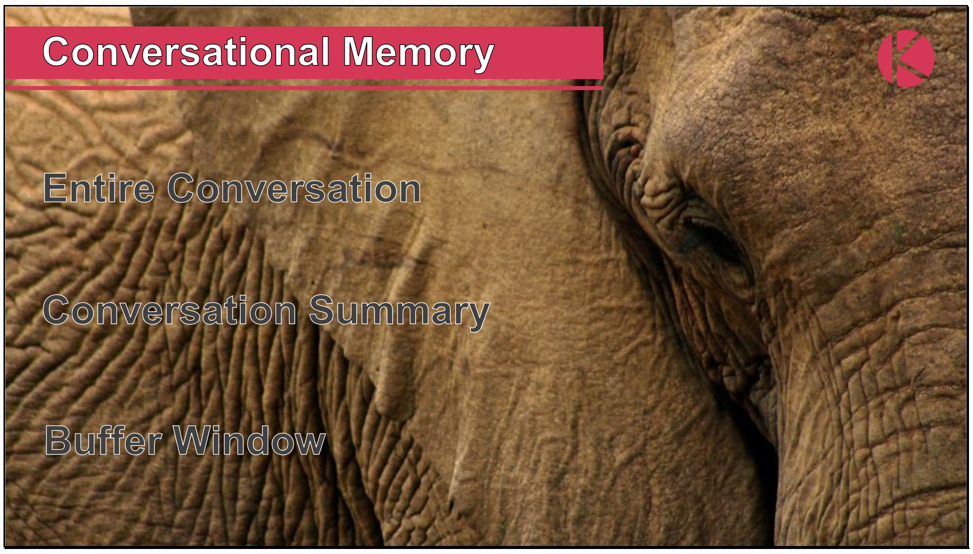
To mitigate these issues, we can implement strategies like Named Entity Recognition. An NER model identifies key entities within text, such as people, organizations, locations, or products, as illustrated by the example on screen. By training a custom NER model, I had a tool to extract more relevant entities like equipment, events, lithology, personnel, and datasets.

The extracted entities can then be used to enhance the query process by tagging these entities onto metadata when adding vectors to the vector database, along with existing metadata like well name, report type, and involved companies. Applying the same process to queries, this allows us to filter data based on the presence or absence of specific entities, like using a "WHERE" clause in SQL. This extra context improves the accuracy and relevance of search results.

<u>Suggested Tools:</u>
Flair
Neural Networks
NLTK
Spacy
Stanford NER Tagger

Brat(Browser-Based Rapid Annotation Tool)
Explosion Prodigy
Label Studio

<u>Additional Areas to Explore:</u>
Topic Modelling



Conversational Memory

Entire Conversation

Conversation Summary

Buffer Window

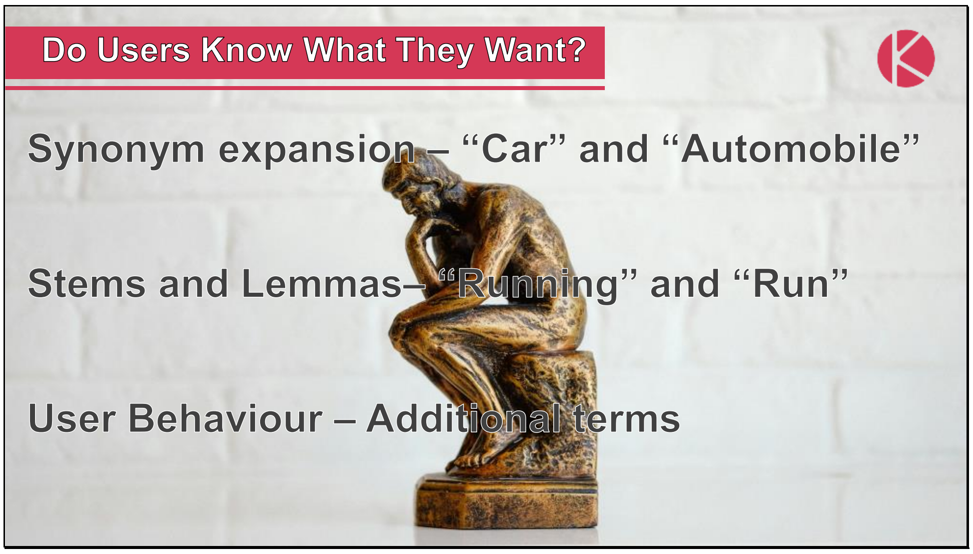
Another issue with LLMs is they are inherently stateless, meaning they treat each query independently. This was inadequate. I needed the system to remember past interactions and use them as context when responding to queries. The system needed short term memory. Luckily, that are several great frameworks out there, I settled on LangChain.

There are several approaches to implementing short term, or conversational, memory. You could store the entire conversation history and pass it to the LLM. This offers maximum data retention, but it can slow down responses, increase costs and limits interaction length due to token/word limits.

Another strategy is to summarise the conversation and pass the summary as context. While this reduces word count and allows for longer conversations, it can significantly increase costs and reliance on the LLM.

I opted for the Buffer Window method. This approach maintains only the most recent interactions, enabling ongoing conversations while keeping costs manageable. However, it may result in some loss of context from earlier parts of the conversation.

Suggested tools:
AutoChain
Custom Self Build
LangChain
LlamaIndex
Vellum



Do Users Know What They Want?

Synonym expansion – “Car” and “Automobile”

Stems and Lemmas– “Running” and “Run”

User Behaviour – Additional terms

Not always.

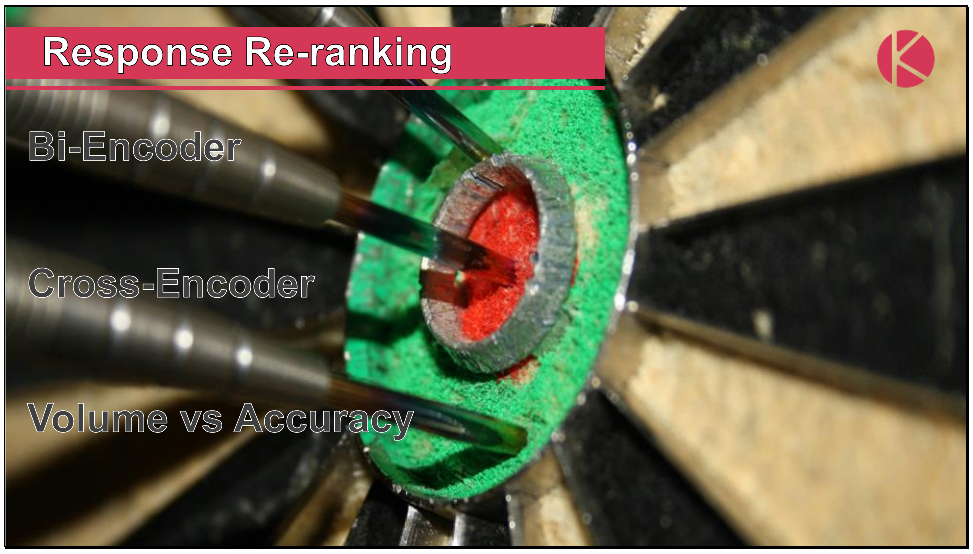
To address vague user queries, we can use Query Expansion, a technique search engines have employed for years to deliver smarter results. It uses natural language processing (NLP) methods like synonym expansion, for example searching for "car" also brings up results for "automobile.“, stemming/lemmatization or using the root of a word, so a search for "running" also includes "run”, and user behavior analysis, for example using data from other users to help suggest additional terms.

Today, query expansion is more sophisticated, lets discuss two common methods:

Pseudo Relevance Feedback (PRF): This method retrieves the top results, scans them for frequently occurring terms (using NER), and adds these to the original query to refine the search. However, if the initial results are off-target it could lead to topic drift, while irrelevant terms might clutter the query, both of which result in the system returning less relevant information.

Large Language Models (LLMs): LLMs infer intent by recognizing subtle cues and context, dynamically expanding queries or suggesting variations. This approach excels at disambiguating terms and handling diverse topics, but it can be resource-intensive, slow, and prone to adding too many terms if not managed well.

I opted to use LLMs because they better capture intent, increasing the chances of retrieving relevant information across a broader spectrum of queries.



Response Re-ranking

Bi-Encoder

Cross-Encoder

Volume vs Accuracy

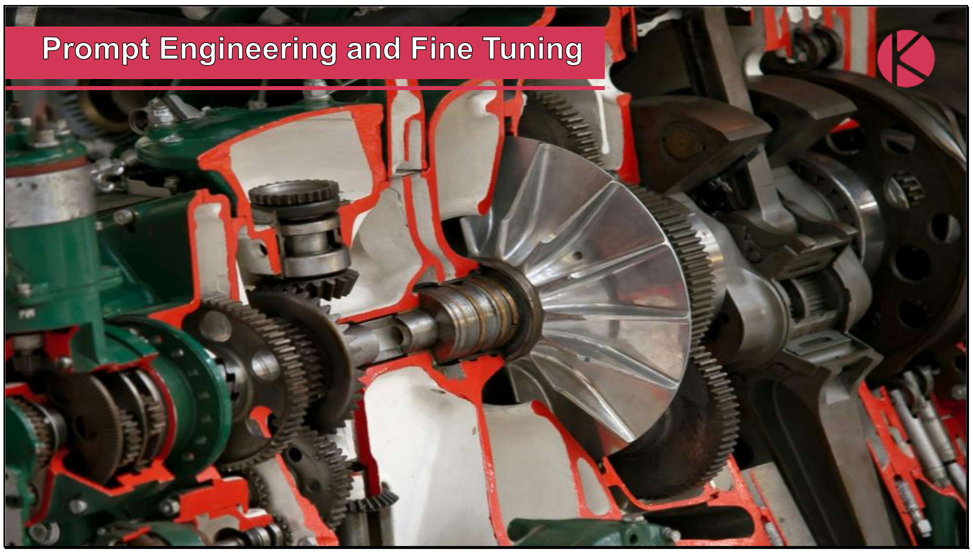
We've expanded our queries to capture user intent, time to refine the system to return only the most relevant results. To achieve this, we need to implement some form of reranking system.

Bi-encoders independently encode the input and output text and then comparing them based on similarity. However, they don't account for the relationships between the texts.

Cross-encoders, on the other hand, use a single encoder to process both elements together, producing a joint representation that captures both the content and relationship between the text. While cross-encoders are slower and more resource-intensive, they offer higher accuracy, especially in cases where subtle differences—like in engineering documents—can significantly impact meaning.

Bi-encoders are more efficient for large-scale comparisons while for smaller datasets, where accuracy is key, cross-encoders are the better choice. Given the limited data being returned, I chose a cross-encoder reranking system to ensure precision and capture those important subtle distinctions, which substantially improved the generated response from the system.

Suggested Tools:
Cohere – Rerank
Custom Self Build
Open_AI cookbook
sentence_transformers from Hugging Face



Prompt Engineering and Fine Tuning

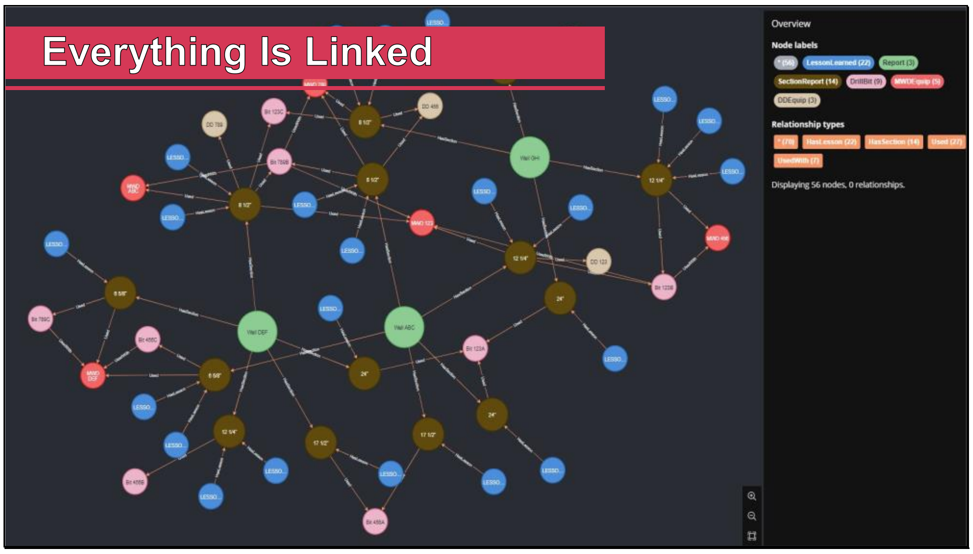
Those familiar with LLMs might be wondering why I haven't mentioned prompt engineering yet. I did incorporate elements of it, such as adjusting queries by assigning specific expert personas—like a drilling engineer, completions engineer, or exploration geologist—depending on the query. I also allowed for fine-tuning of parameters like temperature to control the model's determinism and randomness in generating responses.

However, this is one of the areas where education may be a better option over system controls.

At this point, I switched from ChatGPT to a fine-tuned version of Phi2, a small language model, to have a model focused on the linguistic patterns more prevalent in the Energy Sector. Using the small language model increases control over data security and privacy through hosting in a private cloud environment.

Suggested Tools:
Anthropic – Claude
Cohere - Command
Google – Gemini
Hugging Face – Open-Source Language Models
OpenAI - ChatGPT

CoLab
LoRA
QLoRA



We've made significant progress in optimising the system. But given my background in network graphs, this felt like a perfect opportunity to leverage a graph database, especially as they support vector storage. Looking at the metadata and the NER entities we've extracted I would say there is a basic ontology for the graph. Implementing a graph database we can connect diverse datasets creating a traversable network of data, making it easier to access and link disparate information.

A knowledge graph is a structured data model that organizes and connects data through entities, relationships, and attributes, making complex data easier to analyse and visualize. In the energy industry, knowledge graphs can enhance operations by integrating data from multiple sources—using information such as well locations, geological formations, and equipment—while mapping the relationships between these entities. This interconnected view helps identify patterns and extract insights, improving operational efficiency and decision-making, adding pattern search to the systems formidable arsenal of search tools.

When structured knowledge and its semantics are stored in a knowledge graph, combining semantic matching with structured traversal, the significant potential to enhance human understanding and explainability through data nodes is unlocked, as is machine comprehension through vectors, which enables us to leverage the strengths of both approaches. Knowledge graphs can address challenges RAG faces, such as handling similar documents or information spread across multiple repositories, revealing deeper insights and managing data more effectively. This helps enhance RAG's performance in complex decision-making processes like risk management.

The theoretical Knowledge Graph on the screen illustrates various interconnected elements: Green nodes representing wells, Brown nodes representing drilling sections, Pink nodes for drill bits, Red nodes for MWD equipment, Beige nodes for DD equipment, and Blue nodes for Lessons Learned. By building on this concept, we can continue to add additional nodes and relationships—such as personnel, lithology, wireline logs, mud reports, and NCRs—gradually constructing a comprehensive view of the drilling campaigns and their broader environment.

Suggested Tools:

Amazon Neptune
Graph-tool
Neo4j
NetworkX
TigerGraph

Where Next?

By deploying this system as an API, you can integrate it with platforms like Teams, WhatsApp, Slack, or your own custom interface—or even all of them. The possibilities are only limited by your imagination and ambition.(CLICK)

You could even use Agentic AI to perform actions based on the information retrieved.

What I've described is achievable for any company, regardless of size, using tools like ChromaDB, Neo4j, and Hugging Face.

At this point, you already have a powerful tool. With a user-friendly interface, you can ask questions in natural language, while the underlying knowledge graph connects and organises data from various sources. This opens the door to exciting possibilities like network analysis, centrality techniques, and even edge prediction to discover hidden connections.

There was something my bot was doing consistently that I haven't mentioned yet—it's been including a link to the source data with every response.

I'm going to end on a point I make whenever I speak about AI. If you only take away one thing, I hope it is this: In 2009, Air France flight 447 tragically crashed over the Atlantic, largely due to pilot over-reliance on autopilot. The pilots, though experienced, lacked the hands-on flying skills to manage the crisis. This serves as a critical reminder of the dangers of over reliance on automated.

While AI is a powerful tool, it's essential to maintain our critical thinking and evaluate the responses it provides. Ultimately, we are responsible for how we use it. AI in all its forms is an incredibly powerful efficiency tool, but is only a tool, and ultimately, we have responsibility for that tool and how it is used.

Homework Resources:

Network Science with Python – David Knickerbocker

Text as Data – Brandon M. Stewart

Mastering NLP from Foundations to LLMs – Lior Gazit

Building LLM Powered Applications – Valentina Alto

Graph Machine Learning – Claudio Stamilie

Hands On Large Language Models – Jay Alammar

Advanced Retrieval for AI – deeplearning.ai with ChromaDB

katoni
ENGINEERING