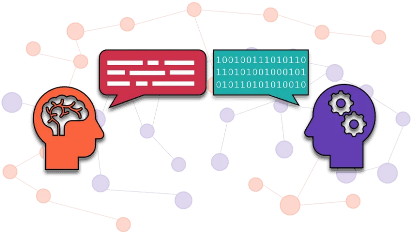
Large Language Models

Konstantinos Skianis, Assistant Professor
Department of Computer Science & Engineering
University of Ioannina, Greece



Makeathon, UniAI
Friday 19/04/2024

1. Introduction
2. Applications and Use Cases
3. Prompt Engineering
4. Retrieval Augmented Generation
5. Summary

Introduction

Large Language Models (LLMs)

Based on the Transformer architecture, which has become the foundation for various state-of-the-art natural language processing (NLP) models

Transformer:

- Attention is All You Need (Vaswani et al., 2017)
- a groundbreaking neural network architecture designed for NLP tasks
- relies on the self-attention mechanism to process and generate sequences
- highly efficient and scalable compared to traditional recurrent neural networks (RNNs) and long short-term memory (LSTM) models

Variants of Transformer Architecture

Various LLMs are built on top of the Transformer architecture with slight modifications or adaptations. Some popular variants include:

- **GPT**: The Generative Pre-trained Transformer (GPT) is an autoregressive model that utilizes only the decoder part of the Transformer architecture to generate text.
- **BERT**: Bidirectional Encoder Representations from Transformers (BERT) is based on the encoder part of the Transformer architecture and is pre-trained using masked language modeling and next sentence prediction tasks.
- **T5**: The Text-to-Text Transfer Transformer (T5) adapts the original Transformer architecture to a unified text-to-text format, enabling it to be used for various NLP tasks with minimal task-specific modifications.

1. Introduction
2. **Applications and Use Cases**
3. Prompt Engineering
4. Retrieval Augmented Generation
5. Summary

Applications and Use Cases of LLMs

- **Sentiment Analysis**: can be used or fine-tuned to analyze the sentiment expressed in a piece of text, determining whether it is positive, negative, or neutral. $\longrightarrow$ gauge customer satisfaction, monitor social media sentiment, or analyze product reviews
- **Question-Answering**: build question-answering systems that provide precise answers to user queries $\longrightarrow$ general or domain-specific
- **Text Summarization**: generate a concise summary of a given input text. This can be useful for creating executive summaries of articles, reports, news stories, or research studies making it easier for readers to quickly grasp the main points
- **Machine Translation**: can be adapted for machine translation tasks, where the goal is to translate text from one language to another. By fine-tuning on parallel corpora, models like T5 have demonstrated state-of-the-art performance in many language pairs

Applications and Use Cases of LLMs (2)

- **Named Entity Recognition (NER)**: can be used to identify and classify named entities (e.g., people, organizations, locations) within a text
  → information extraction, content recommendation, and semantic search
- **Text Classification**: classify text into various categories
  → topic classification, spam detection, or document tagging content filtering and recommendation systems
- **Text Generation**: LLMs, especially autoregressive models like GPT, can be used for text generation tasks
  →generate coherent and contextually relevant text based on a given input, which can be useful for tasks like chatbot development, content creation, or creative writing prompts
- **Few-Shot Learning**: can adapt to new tasks with minimal task-specific examples → enable more efficient transfer learning and reduce the need for extensive fine-tuning

1 Introduction

2 Applications and Use Cases

3 Prompt Engineering

4 Retrieval Augmented Generation

5 Summary

Prompt Engineering

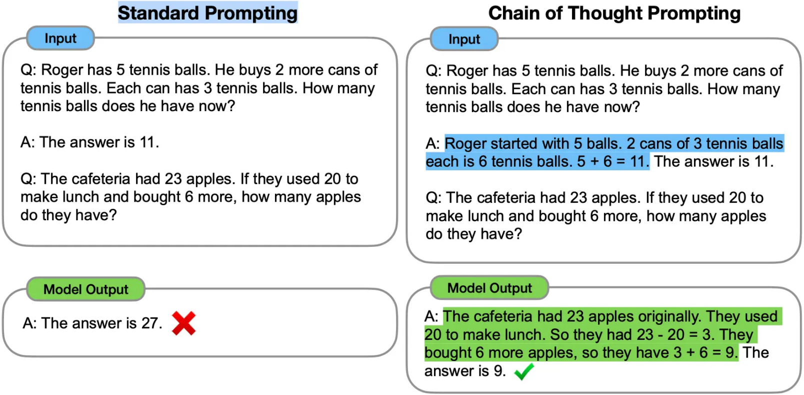
(1) Explicitness: specify the format you want:
- Less explicit prompt: Translate the following English text to French: "Hello, how are you?"
- More explicit prompt: Translate the following English text to French and provide the translation in quotes: "Hello, how are you?"

(2) In-context examples: provide examples within the prompt to guide the model towards generating responses in the desired format or style:
- Without example: Write a haiku about nature.
- With example: Write a haiku about nature, following the 5-7-5 syllable pattern. For example: "Gentle morning rain, / Caressing the earth below, / Nature's symphony."

(3) Limit response length: specify a maximum response length to prevent excessively long or verbose answers.
- Without length limit: Explain the greenhouse effect.
- With length limit: Explain the greenhouse effect in less than 3 sentences

(4) Requesting rationales or step-by-step explanations:
- without rationale: What is the square root of 16?
- with rationale: What is the square root of 16, and explain why that is the correct answer

Chain-of-Thought



Standard Prompting

Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ✗

Chain of Thought Prompting

Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9. ✓

1 Introduction

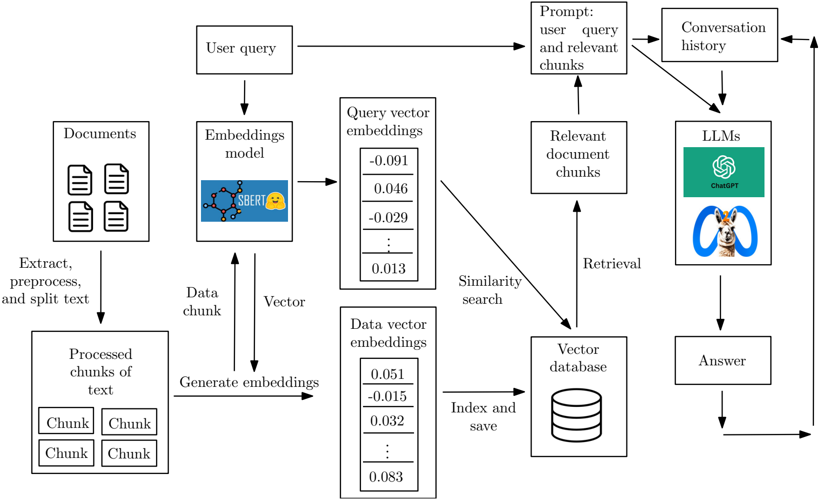
2 Applications and Use Cases

3 Prompt Engineering

4 Retrieval Augmented Generation

5 Summary

Retrieval Augmented Generation (RAG) Pipeline



A full schematic illustration, based on: https://ubuntu.com/blog/llms-explained

Why use RAG?

Benefits

- **Consistency:** more likely to get the same answer from the same question
- **Increased factual accuracy:** LLM's responses are based on the provided information → model is less likely to hallucinate or mislead
- **Decreased cost:** building a RAG pipeline is less expensive than fine-tuning. Update only the database instead of training a new model
- **Currency:** ensure the LLM's responses are based on up-to-date data
- **Easily verifiable source:** access to the source for cross-checking
- **Strong scalability:** millions of documents

Cons

- **Complexity:** chunking, quality of search, low-quality data
- **Context length limitation:** ChatGPT (gpt-3.5-turbo) has a maximum context length of 15k tokens
- **Limited creativity:** losing the point of generative AI

1. Introduction
2. Applications and Use Cases
3. Prompt Engineering
4. Retrieval Augmented Generation
5. **Summary**

Summary

We talked about:

- LLMs and applications
- Prompt engineering
- Retrieval Augmented Generation

Link: https://github.com/cohere-ai/notebooks/blob/main/notebooks/Vanilla_RAG.ipynb

Coding time!