Large Language Models and Generative AI for NLP - Lecture 2

Aarne Talman

Machine Learning & Data Science Lead, EMEA Centre for Advanced AI at Accenture
Visiting Researcher and Lecturer at University of Helsinki

Contents

1. Recap of lecture 1
2. Introduction to prompting
3. Prompt engineering techniques
4. Hands-on examples
5. Lab

Contents

Syllabus

Week 1: Introduction to Generative AI and Large Language Models (LLM)

- Introduction to Large Language Models (LLMs) and their architecture
- Overview of Generative AI and its applications in NLP
- Lab: Tokenizers

Week 2: Using LLMs and Prompting-based approaches

- Understanding prompt engineering and its importance in working with LLMs
- Exploring different prompting techniques for various NLP tasks
- Hands-on lab: Experimenting with different prompts and evaluating their effectiveness

Week 3: Evaluating LLMs

- Understanding the challenges and metrics involved in evaluating LLMs
- Exploring different evaluation frameworks and benchmarks
- Hands-on lab: Evaluating LLMs using different metrics and benchmarks

Week 4: Fine-tuning LLMs

- Understanding the concept of fine-tuning and its benefits
- Exploring different fine-tuning techniques and strategies
- Hands-on lab: Fine-tuning an LLM for a specific NLP task

Week 5: Retrieval Augmented Generation (RAG)

- Understanding the concept of RAG and its advantages
- Exploring different RAG architectures and techniques
- Hands-on lab: Implementing a RAG system for a specific NLP task

Week 6: Use cases and applications of LLMs

- Exploring various real-world applications of LLMs in NLP
- Discussing the potential impact of LLMs on different industries
- Hands-on lab: TBD

Week 7: Final report preparation

- Students work on their final reports, showcasing their understanding of the labs and the concepts learned.

What are language models?

- **Language modeling** is the task of **estimating the probability distribution of sequences of words or tokens in a given language**.
- More formally, given a sequence of words $\mathbf{w_1 w_2 \dots w_n}$ in a language ***L***, a language model assigns a probability $\mathbf{P(w_1 w_2 \dots w_n)}$ to the entire sequence.
- This can also be expressed as the **conditional probability of the next word given the previous words**:

$$\mathbf{P(w_n \mid w_1 w_2 \dots w_{n-1})}$$

- In essence, **language modeling aims to capture the statistical patterns and structures of a language**, allowing it **to predict the likelihood of a given sequence of words** or to **generate new text** based on existing sequences in the language.

Transformer models

Self-Attention mechanism:

- The heart of the Transformer, enabling it to **weigh the importance of different words in a sequence** when making predictions.

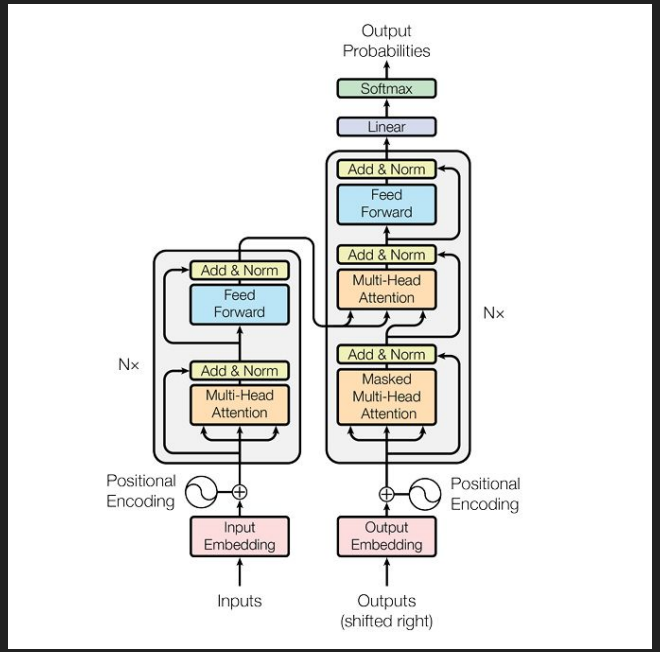
Multi-Head Attention:

- Employs **multiple attention heads in parallel,** each focusing on different aspects of the input sequence.

Positional Encoding:

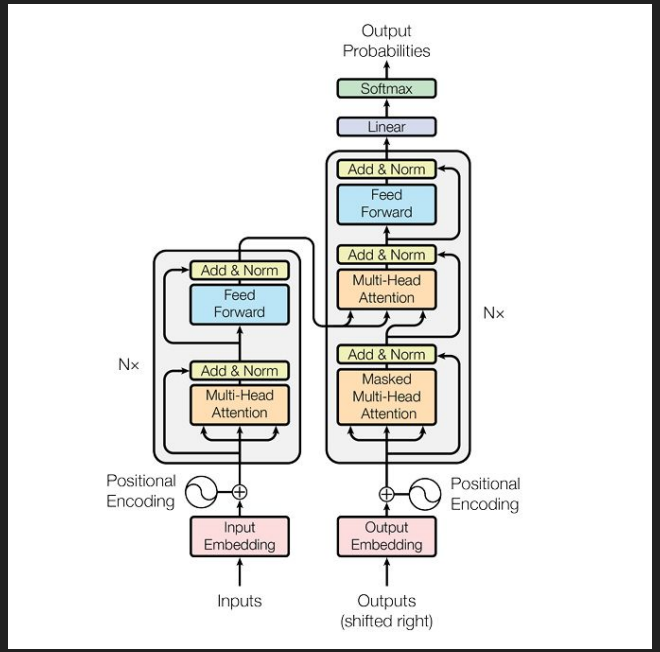
- Positional encodings are added to the input embeddings, providing the model with **information about the position of each word in the sequence**.

Encoder-Decoder (T5, NMT models), **Encoder only** (BERT, RoBERTa, Sentence Transformers), **Decoder only** (GPT models, LLaMa, Mistral etc.)



Transformer Architecture

- **Captures long-range dependencies:** The self-attention mechanism allows the model to effectively capture relationships between words that are far apart in the sequence, overcoming limitations of previous models like RNNs.
- **Parallel processing:** The Transformer architecture is highly parallelizable, enabling faster training and inference on modern hardware.
- **Strong performance on various NLP tasks:** Transformers have achieved state-of-the-art results on a wide range of NLP tasks, including machine translation, text summarization, question answering, and many others.



Source: Vaswani et al. 2017

Contents

Pre-training

Large language models, like GPT-3, have been trained on massive amounts of textual data scraped from the internet.

In the process they have not only learned to predict the next tokens, but also various associations regarding:

- **Style:** e.g. formal text, informal text, etc.
- **Tone:** e.g. polite, direct, etc.
- **Domain:** e.g. medicine, finance, sports, etc.
- …

Accessing “emergent capabilities” through prompting

- The GPT-3 paper **"Language Models are Few-Shot Learners"** (Brown et al., 2020) was one of the first papers to argue that **prompting is a powerful technique for leveraging the vast knowledge encoded in large language models**.
- By providing the **right context and examples**, we can **guide these models to perform a wide range of tasks** with remarkable flexibility and efficiency.
- They argue that GPT-3 has gained **“emergent” capabilities**:
  - capabilities that the model was not explicitly trained to have but that have emerged through pre-training.
- They also argue that the model has **learned to learn new tasks from just a few examples**

What is prompting?

Prompting is essentially **providing instructions or inputs to a pre-trained large language model** to guide its output.

Prompts can take many forms:

- **Instructions:** "Write a short story about a cat who goes on an adventure."
- **Questions:** "What are the main causes of climate change?"
- **Examples:** Providing a few examples of the desired output format.
- **Constraints: "**Translate this paragraph into French, using formal language."

Why is prompting important?

Prompts guide the model towards the desired output.

- **Specificity:** By tailoring prompts to specific tasks, we can **leverage the emergent capabilities of LLMs** for a wide range of applications, from writing poems to generating code.
- **Control:** Prompts allow us to control various aspects of the output, such as length, style, and format.
- **Efficiency:** Prompting enables us to adapt LLMs to new tasks quickly without retraining the entire model.
- **Creativity:** Well-crafted prompts can enable LLM's to be more “creative” and lead to surprising and insightful outputs.

Contents

Prompting techniques

Zero-shot prompting:

- Leverage the skills learned during pre-training by providing a simple instruction or question to the model. E.g.
  - **Prompt:** *“Classify the text into neutral, negative or positive. Text: I think this burger is great. Sentiment:”*
  - **Answer:** *“Positive”*

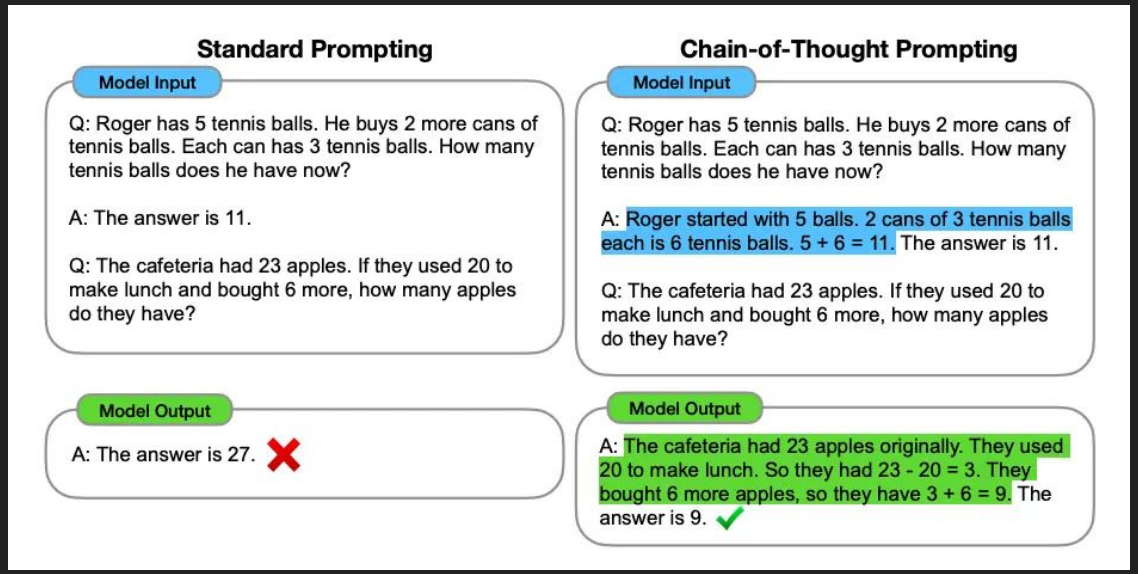
Few-shot prompting:

- Provide examples of desired output before asking the question or instruction. E.g.
  - **Prompt:** *“Answer to with one word. I like basketball: negative; I hate school: positive; I love hamburgers: negative; I hate work:”*
  - **Answer:** *“positive”*

Prompting techniques

Chain-of-thought (CoT) prompting Wei et al. 2023, “Chain-of-Thought Prompting Elicits Reasoning in Large Language Models”

- CoT enables complex reasoning capabilities through intermediate reasoning steps



Prompting techniques

Prompt Chaining

- Combine multiple prompts together, breaking down a task into smaller subtasks.
  - Example:
    - **Prompt 1:** *"Generate a list of 5 creative story ideas involving a talking dog."*
    - **Prompt 2:** *"Write a short story based on the third idea from the following list: [insert output from Prompt 1]. The story should be written from the dog's perspective and have a humorous tone."*
  - Example:
    - **Prompt 1:** "Write Python code to generate a list of the first 10 Fibonacci numbers."
    - **Prompt 2:** "Modify the following Python code to print the Fibonacci sequence in reverse order: [insert output from Prompt 1]"

Prompting techniques

Other techniques:

- **Tree-of-thought prompting:** encourages LLMs to break down complex problems into smaller steps and explore multiple reasoning paths, much like how humans think.
  - E.g. Ask the model to provide alternatives, then ask it to evaluate these alternatives and select the best one for the task.
- **Asking the model to generate a prompt for the task:**
  - **Example:**
    - **Prompt 1:** *“Generate a prompt that will instruct an LLM to write a short story about a cat who travels through time. The prompt should include 3 few-shot examples of short stories with different themes, and it should specify that the story should be written in a whimsical tone.”*
    - **Prompt 2:** *<insert output of the first prompt>*

In-Context Learning

- In-context learning is a generalisation of few-shot learning where the **LLM is provided a context as part of the prompt** and asked to **respond by utilising the information in the context**.
  - **Example:** *"Summarize this research article into one paragraph highlighting its strengths and weaknesses: <insert article text>"*
  - **Example:** *"Extract all the quotes from this text and organize them in alphabetical order: <insert text> "*
- A very popular technique that you will learn in week 5 called **Retrieval-Augmented Generation (RAG)** is a form of in-context learning, where:
  - **a search engine** is used to **retrieve some relevant information**
  - **that information** is then provided to the LLM as **context**

Prompt templates

- Prompt templates are a convenient way to create reusable templates that contain certain elements that can be modified while keeping the structure of the prompt fixed.

- **Template:** *“Summarize the following text about {topic} in {number} bullet points: {text}”*
- **Parameters:**
  - {topic}: The subject of the text to be summarized.
  - {number}: The desired number of bullet points in the summary.
  - {text}: The text that needs to be summarized.

Contents

Contents

Preparations for the lab

- **Gemini API Key:**
  - Use your existing google account or create a new free account
  - Go to https://aistudio.google.com/apikey
  - Get your API key and store it in a safe place
- **Create a HuggingFace account:**
  - https://huggingface.co/join
- **Set up Python & Jupyter environment:**
  - I use Python 3.12 for running code locally
  - For inference or fine-tuning requiring GPU, I use Google Colab (more information later)

Lab exercise

1. For this exercise use the **gemini-chatbot** and/or **prompting-notebook** found in GitHub under week-2 folder.
    a. Select a domain (e.g. finance, sports, cooking), tone of voice, style and persona (e.g. a pirate) and a question/task you want to accomplish (e.g. write a blog post)
    b. Modify the gemini-chatbot and test the different prompting approaches discussed in the lecture to achieve the task.
    c. Do the same for **prompting-notebook** (run this in Google Colab using a T4 GPU backend)
    d. Write a section to your report explaining what you did and what were your findings. Which prompting approach worked the best and why?
2. Modify the **in-context-learning** notebook (you can run this locally or in Google Colab)
    a. Modify the prompt to change the style of the output to be a table with strengths and weaknesses in separate columns. (Markdown printing should show the table correctly. If you have time, modify the html printing to show the updated style as a table).
    b. If you have time: modify the notebook to use an open source model from Hugging Face instead of Gemini

References for further study

- Brown et al. 2020, Language Models are Few-Shot Learners
- Wei at a. 2023, Chain-of-Thought Prompting Elicits Reasoning in Large Language Models
- Prompt Engineering Guide
- Gemini docs