Large Language Models

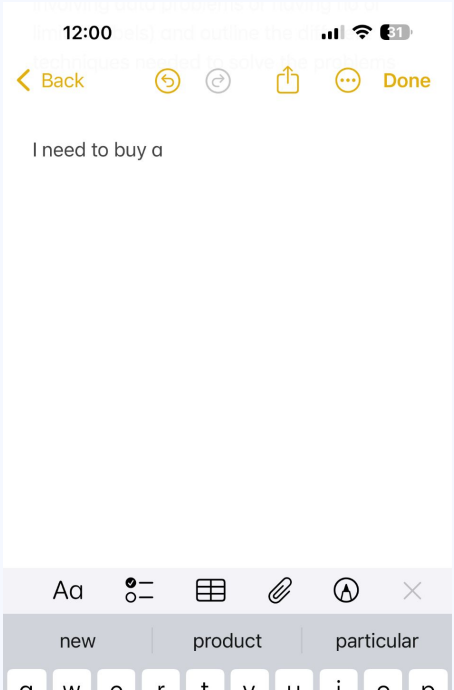
Here is where your presentation begins

Large Language Models

- Have you ever wondered how deepseek, chatgpt, and other AI models work?

Large Language Models

- They are like a smart autocomplete – predicting the next token given a sequence of tokens coming beforehand



Large Language Models

- When it receives a prompt, it writes the first word by classifying the most likely next token out of all possible tokens
- So the output size for each forward pass of the language model is its vocabulary size

Large Language Models

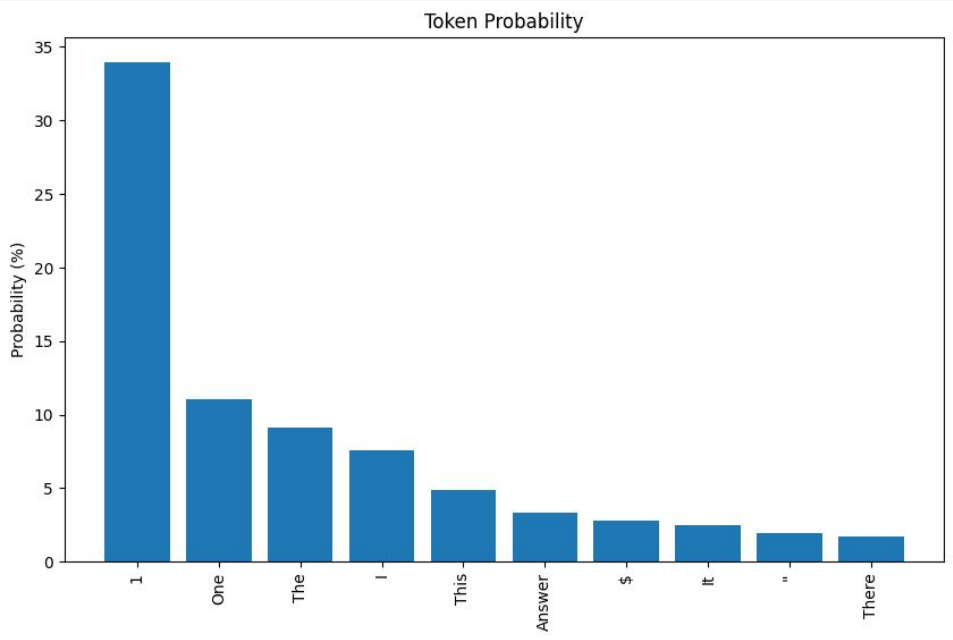
The prediction process repeats, as it feeds its own prediction into the context from which it predicts again.

- This is called autoregressive prediction

For example:
Input: We; output: had
Input: We had; output: We had a
Input: We had a; output: We had a good
Input: We had a good; output: We had a good experience

Large Language Models

- The model outputs a probability distribution over all possible tokens (by predicting real numbered logit scores, then applying softmax to them)



Decoding

- How do you actually choose which token to output?
- Intuitively it would be the most likely token predicted, as it would be for other classification tasks

Decoding

- But in reality, the model might become too boring and not creative enough, or even output suboptimal solutions!
- E.g., if the context contains “I have a”, there are many possibilities on what comes next, but with greedy decoding (only choosing the most likely token), it can only output one fixed result every time
- It needs a way to choose when to output less likely tokens…

Sampling

- We sample from the probability distribution
- For example, if the probability that “Paris” follows “the capital of France is” is 99%, 99 times out of 100 when the language model is fed “the capital of France is”, it would output “Paris” as the next token.

Hallucination

- This is where hallucinations come from
- Especially if the model is not certain (does not have the required knowledge), the option of outputting a token that appears to be relevant (not something like “I don’t know”) is still significant
- So by chance, it would end up predicting a factually wrong token and keep going on that path of token predictions to provide wrong statements!

Hallucination

- For example, if the probability that “Paris” follows “the capital of France is” is 99%, there’s a 1% chance that the model outputs something else, which is wrong as “Paris” is the only correct answer.

Hallucination

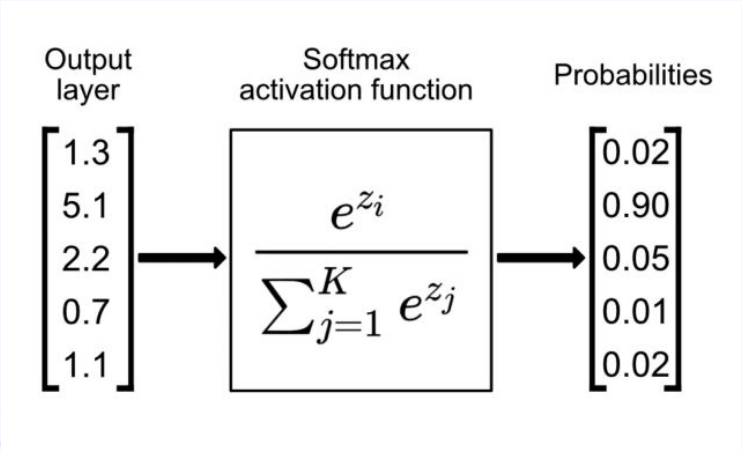
- AI models do not explicitly care about whether their predictions are factual or not. They only care about what are statistically the most likely paths forward
- Moreover, errors compound after one hallucination occurs: it generates tokens with the factually wrong previous token(s) in mind to ensure a coherent flow of ideas, making it keep generating the wrong information

Sampling

- But what if a fixed softmax function is also too constrained?
- That’s where the world of sampling parameters come in.
- Temperature, top-k, and top-p…

Temperature

- In classification (which is what the model does for each token), raw logit scores are transformed into probabilities via softmax



Temperature

- The resulting probability distribution varies with the scale of the raw input logits!
- For example, applying softmax to [1, 2, 3] returns a different distribution from applying softmax to [2, 4, 6]

Temperature

- Therefore, we can actually use a hyperparameter called temperature to adjust the scale of the raw logits before the softmax
- Lower temperature (close to 0.0) means more deterministic output (the probability distribution is dominated by the largest logit score), while higher temperature “evens out” the probability distribution, assigning higher probability to lower logit scores

$$\frac{exp(z_i/T)}{\sum_j exp(z_j/T)}$$

Temperature

- Example: your model can output the words “I” (class 0), “have” (class 1), and “a” (class 2). When fed the sentence “This is”, the model outputs logit scores of [-0.2, -0.3, 1.6].
- a) If the model has temperature 0.1, what is the probability assigned to “a” (class 2)?
- Answer: exp(1.6 / 0.1) / (exp(-0.2 / 0.1) + exp(-0.3 / 0.1) + exp(1.6 / 0.1))
  = 0.99999997916
- Result: the model almost always selects “a”.

Temperature

- Example: your model can output the words “I” (class 0), “have” (class 1), and “a” (class 2). When fed the sentence “This is”, the model outputs logit scores of [-0.2, -0.3, 1.6].
- b) If the model has temperature 4.0, what is the probability assigned to “a” (class 2)?
- Answer: exp(1.6 / 4.0) / (exp(-0.2 / 4.0) + exp(-0.3 / 4.0) + exp(1.6 / 4.0))
  = 0.442573204
- Result: the model selects “a” only 44% of the time. It selects either “I” or “have” 56% of the time, both of these answers being grammatically incorrect.

Temperature

- For most applications, temperature ranges from 0.0 to 1.0; it is almost never set to above 1.0, like in the example above.
- For factual recall, set the temperature to lower values (below 0.5) to increase the probability of outputting the most likely, correct tokens (if the model is well-trained enough)
- For creative tasks, set the temperature to higher values (above 0.5) for the model to select less probable tokens, allowing a more diverse range of possible ideas

Top-p and Top-k

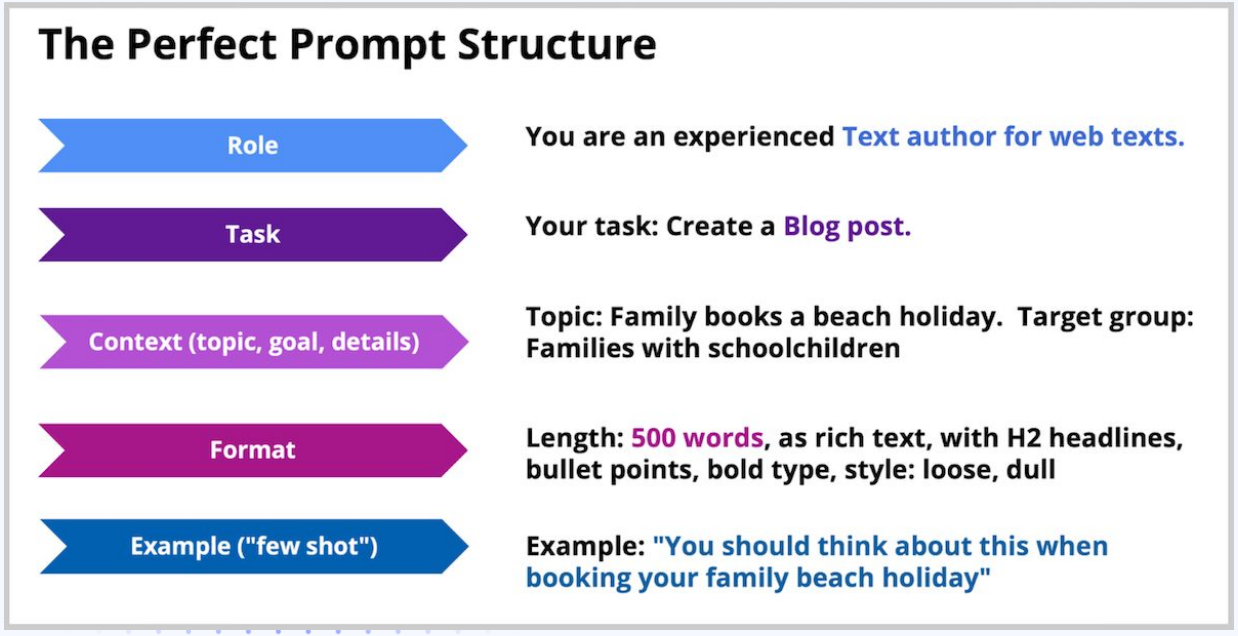
- These settings provide filters that select only the most probable tokens
- Top-k: sample from only the top k tokens
- Top-p: sample from only the top k tokens which have a cumulative probability of p

Try It Out!

- Tell an LLM to write something!
  https://colab.research.google.com/drive/1wtQ3zs0W_TnQcL123trQjdijAHDMf3W-
- Adjust the temperature, prompt, and system prompt and see how it goes

Prompt

- The message that you give to the AI model for it to generate stuff



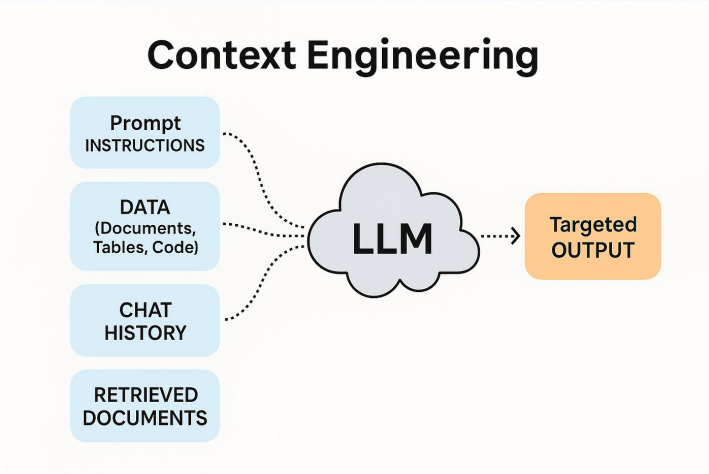
Prompt

- Split into the system prompt and the user prompt
- The system prompt dictates the model's generation style and ground rules that the model must follow, and is usually in the settings section or hidden from view in commercial LLM interfaces
- The user prompt is what you actually ask the model to do

Criteria	User prompts	System prompts
Purpose	Task-specific instructions	Overall framework & guidelines
Frequency of use	Used frequently, often changed	Set once, rarely changed
Scope	Narrow, focused on individual tasks	Broad, applies to all interactions
Content focus	Specific details, context & desired outcomes	General rules, tone, ethics & brand values
Example	"Write a follow-up email to prospect X about Y product"	"You are a seasoned account executive for a B2B SaaS company..."
Typical length	Short to medium (1-5 sentences)	Medium to long (paragraph or more)
Primary impact	Output content & structure	Overall tone, behavior & approach
When to use	For each specific task or request	At the beginning of AI solution setup or a new session
Modifiability	Easily modified for each new task	Requires careful consideration to change

Context

- Everything that the LLM receives, including your system prompt, your prompt, and the model's responses in the preceding turns of the conversation
- Can also include documents retrieved by RAG systems and summaries of previous chats with the same user
- Wider coverage than just asking the question, as you can provide information to base the LLM's response on



- **Instructions / System Prompt:** An initial set of instructions that define the behavior of the model during a conversation, can/should include examples, rules ....
- **User Prompt:** Immediate task or question from the user.
- **State / History (short-term Memory):** The current conversation, including user and model responses that have led to this moment.
- **Long-Term Memory:** Persistent knowledge base, gathered across many prior conversations, containing learned user preferences, summaries of past projects, or facts it has been told to remember for future use.
- **Retrieved Information (RAG):** External, up-to-date knowledge, relevant information from documents, databases, or APIs to answer specific questions.
- **Available Tools:** Definitions of all the functions or built-in tools it can call (e.g., check_inventory, send_email).
- **Structured Output:** Definitions on the format of the model's response, e.g. a JSON object.

Context

- There is a maximum number of tokens that can be passed through a language model
- This number, the context length, is chosen when the model is being pretrained on large amounts of text on the internet

Context

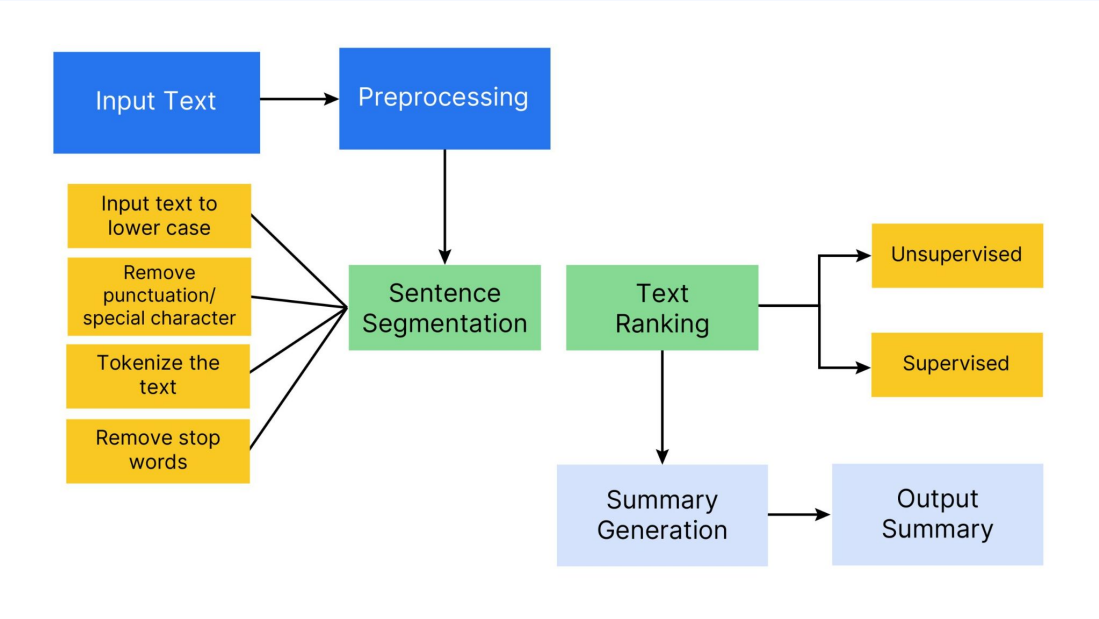
- The transformer architecture, which is what most modern LLMs are based on, have a time and space complexity of O(n^2) where n is the number of input tokens.
- This means the compute and memory resources needed explode quadratically relative to the number of tokens!

Context

- Context length depends on the model architecture and the hardware resources it’s designed to run on
- Smaller models can only handle a few thousand tokens at a time, while state-of-the-art commercial models can handle upwards of a million tokens of context

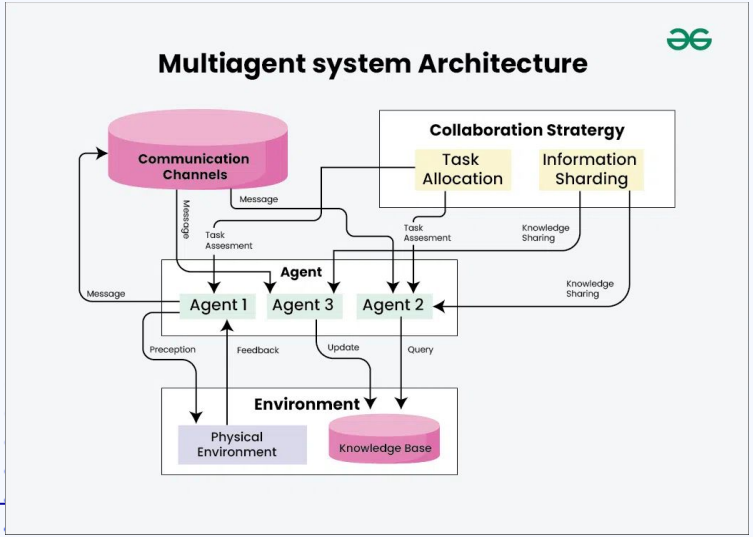
Context

- To overcome context length limitations, truncate the text (so that only the latest context_length tokens are kept), or summarize it into a shorter text before generating tokens again



Agents

- A lot of complete AI system designs include the concept of agents
- Each AI agent does a specific task like text retrieval, OCR (optical character recognition), object detection, or text summarization
- They communicate with each other to provide information for the main pipeline (e.g., the orchestrator, which manages the agents) to act on



Agents

- Advantages:
- enable AI models to collaborate to improve efficiency / accuracy
- enables the use of smaller models for specialized or routine tasks, which can be faster and less costly than letting one large model handle everything
- provide slightly more information when debugging erroneous responses by isolating the problem to one of the components and then improving it

Agents

- Disadvantages:
- hallucinations can compound as the models pass the wrong information to each other, leading ultimately to wrong or irrelevant responses
- it could be challenging to maintain the complicated agentic pipelines
- it could be challenging to ensure low latency especially if different agents communicate with each other through the network / if different service providers are responsible for different models