CSC-496/696: Natural Language Processing and Text as Data

Lecture 19: Prompt Engineering II

Patrick Wu

Tuesday, November 12, 2024

Lecture Contents

1. Announcements
2. Review of Prompt Engineering
3. Hyperparameters When Using Generative LLMs Without Gradient Updates
4. Retrieval-Augmented Generation
5. Lab

Announcements

Assignments

- Assignment 4 today at 11:59pm
- Last assignment will be released today
  - One problem, but several parts
- Graded Assignment 3s coming soon!

Questions?

Review of Prompt Engineering

Prompting

- **Prompts**: instructions we give to a language model in natural language
- Basis of prompting is on *contextual generation*
- Given the prompt as context, the LLM will generate the next token based on its token probability, conditioned on the prompt: $P(w_i|w_{<i})$

Recap on Training Generative LLMs

- Next word prediction: given a sentence, we predict the next word
- "The next four years will be [MASK]"

Prompting: In-Context Learning

- Prompting can also be viewed as a **learning signal**
- It has been shown that having demonstrations in a prompt (examples to help make the instructions clearer) or including definitions can improve answers
- This is called **in-context learning**: the model is learning from information the prompt even though there are no gradient-based updates to the model's underlying parameters

Instruction Finetuning and Preference Alignment

- Jointly called "model alignment"
- Uses human-labeled data to fine-tune the model to follow directions and respond in a way that humans would "prefer"
    - "Human preference" a nebulous concept here

Instruction Finetuning

- Used to train the model to follow instructions well
- Gets us away from having to use sentence completion to use generative LLMs
- Back in my day, you had to use the sentence completion logic to get the generative LLM to do something useful
- For example, a prompt used for classification could be

  Tweet: [insert tweet here]

  The sentiment of this tweet is
- Because the generative LLM is autoregressive, it will generate the next words to finish this sentence

Human Feedback

- We also want the model to respond in certain ways
- We don't want the model, for example, to give us instructions to do something bad
- But it's also time-consuming (and very ill-defined) to ask people to create "harmless" and "helpful" text
- Unlike instruction tuning, we don't really have a good way to create generalizable examples
- Instead, we use human feedback from the text the model generates to improve the model
- In practice, this comes in the form of ranking the model's outputs

Example of a Reward Model

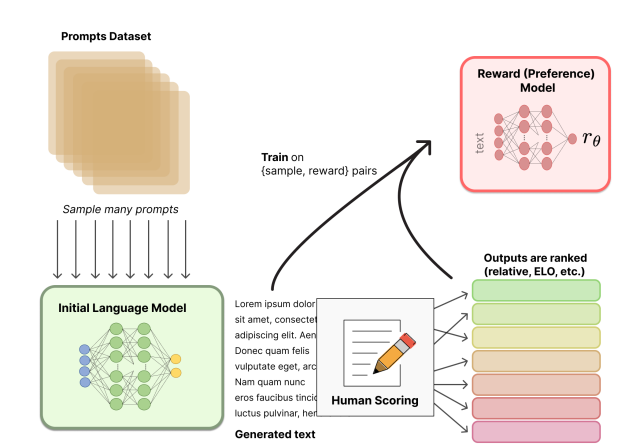


Image from Hugging Face

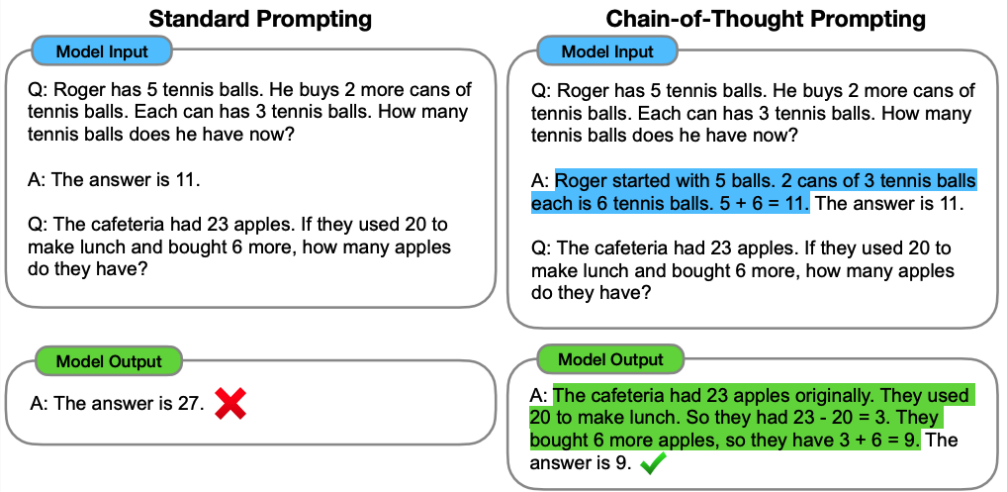
Zero-Shot vs. Few-Shot Prompting

- Zero-Shot Prompting: prompt used to interact with the model won't contain examples or demonstrations
- Few-Shot Prompting: prompt used to interact with the model that does contain a few examples or demonstrations
  - Can be as little as one demonstration
  - Often works well when dealing with abstract concepts

Few-Shot Prompting

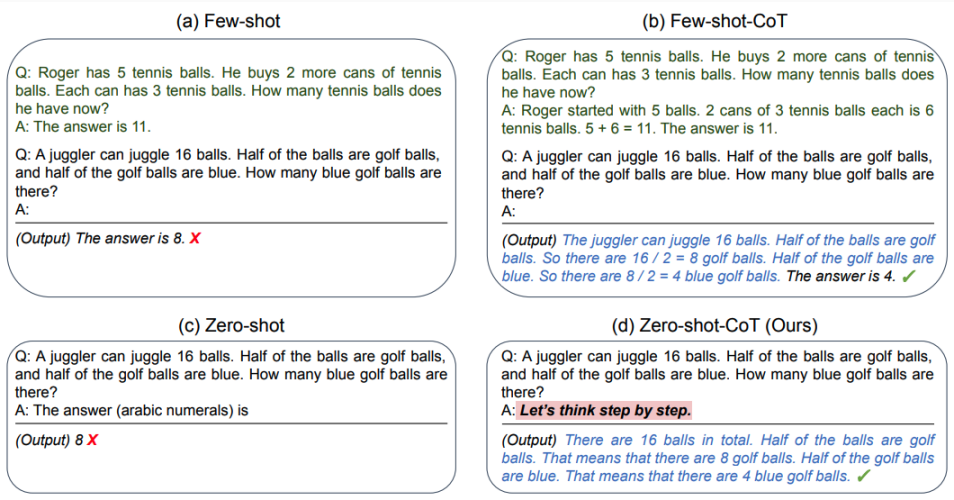
- Can often overfit to your examples
- What *we* think are good examples might actually not be too helpful for the LLM
- Suggestion is to try both zero-shot and few-shot prompting and see which one does better

Chain of Thought Prompting



From Wei et al. (2022)

Zero-Shot Chain of Thought Prompting



From Kojima et al. (2022)

Review Question 1

Zero-shot prompting involves using exemplars or demonstrations

(A) True

(B) False

Review Question 2

Why do we use RLHF instead of directly finetuning the model as we do with instruction finetuning?

(A) We want to make things harder for ourselves

(B) It is difficult to generate examples of “good” or “bad” texts

(C) It is more efficient than supervised finetuning

(D) It requires less human labeling

Review Question 3

A generative large language model trained on next-token prediction can be used with instructions.

(A) True

(B) False

Hyperparameters When Using Generative LLMs Without Gradient Updates

Sampling for LLM Text Generation

- When the generative model generates a new word, it chooses the next word based on the probability distribution
- Sampling from a model's distribution means choosing a random word according to the probability assigned by the model
- For example, "yes" might have a 60% chance of being selected and "no" has a 40% chance of being selected
- It turns out that this approach doesn't work too well...if you end up randomly choosing a rare word, the next words might be even rarer
- You might end up generating nonsense

Sampling Methods

- But we still want to make sure that our sampling methods can generate both *high quality* text and a *diversity* of text
- We typically control this through two parameters: top-p sampling and temperature sampling

Top-p Sampling

- We can keep the top p percent of the probability mass and then select a word among the kept words
- Technically, it is defined as: given the distribution $P(w_t|\mathbf{w}_{<t})$, the top-p vocabulary $V^{(p)}$ is the smallest set of words such that

$$\sum_{w \in V^{(p)}} P(w|\mathbf{w}_{<t}) \geq p$$

Top-p Sampling

Let's say that the probabilities of the next words are as follows:

- **cat**: 0.4
- **dog**: 0.3
- **bird**: 0.2
- **turtle**: 0.1

If we go down the list, the cumulative probabilities are

- "cat" $\rightarrow$ cumulative probability: 0.4
- "dog" $\rightarrow$ cumulative probability: 0.7
- "bird" $\rightarrow$ cumulative probability: 0.9
- "turtle" $\rightarrow$ cumulative probability: 1.0

Top-p Sampling

If we set our `top_p` parameter to 0.6, then the model will only sample from "cat" and "dog" in this scenario

This is the smallest set of words such that $\sum_{w \in V^{(p)}} P(w|\mathbf{w}_{<t}) \geq 0.6$

Top-p Sampling

If we set our `top_p` parameter to 0.6, then the model will only sample from "cat" and "dog" in this scenario

This is the smallest set of words such that $\sum_{w \in V^{(p)}} P(w|\mathbf{w}_{<t}) \geq 0.6$

The probabilities of the words are then normalized

- **cat**: $\frac{0.4}{0.7} = 0.57$
- **dog**: $\frac{0.3}{0.7} = 0.43$

This will then form the basis of our new sampling approach

Top-p Sampling

Notice that this does two things

- It aims to keep the high-quality words
- Reduces diversity, but does not impose a hard cutoff on the number of words that can be sampled
- If a bunch of words have smaller probabilities, then the model will still be able to choose among a large pool of words
- If only a few words have high probability, one of those words will almost certainly be chosen

Temperature Sampling

- In temperature scaling, we don't truncate the distribution; we just reshape it
- First, define $\mathbf{u}$ as the logit (the raw number that comes out of the LLM) of all tokens
- We usually calculate probabilities by using

$$\mathbf{y} = \text{softmax}(\mathbf{u}))$$

- But we can use the temperature parameter, τ, to change these probabilities

$$\mathbf{y} = \text{softmax}(\mathbf{u}/\tau)$$

Temperature Sampling

- If $\tau \in (0, 1]$, applying the softmax to $\mathbf{u}/\tau$ will **increase the probabilities of the tokens that already have a high probability** and **decrease the probabilities of the tokens that already have a low probability**
- In other words, it makes the distribution spikey
- If $\tau > 1$, it **flattens the probability distribution**
- This is known as *high-temperature sampling*

Temperature Sampling

- The temperature parameter effectively regulates the amount of randomness in the generated output, which can lead to more diverse outputs
- It is often called the "creativity" parameter, although this isn't quite right
  - Randomness $\neq$ creativity

Temperature Sampling and Top-p Sampling

- For almost all LLMs, you can control these hyperparameters
- For example, you can control these parameters if you use the API version of OpenAI's GPT models, but you cannot control these parameters if you use the web GUI version (the one you can access at `chatgpt.com`)
- You can change both the temperature sampling and the top-p sampling; some guides, such as those by OpenAI, suggest that you only change one
- In practice, the temperature (τ) is often changed

Default Values

- It's important to know what the default values are for each model
- For example, GPT uses a temperature of 1 and a top-p sampling rate of 1 as the default
- GPT allows the temperature to go all the way up to 2, with values greater than 1 effectively flattening the probability distribution
- Llama-3.2-1B-Instruct, on the other hand, uses a default temperature parameter of 0.6 and a top-p rate of 0.9 (see here: https://huggingface.co/meta-llama/Llama-3.2-1B-Instruct/blob/main/generation_config.json)

Review Question 4

If the temperature parameter τ is greater than 1, it makes the probability distribution more spikey (i.e., large probabilities get even larger, and small probabilities get even smaller).

(A) True

(B) False

Review Question 5

What is the objective of top-p sampling?

(A) Decreases the number of vocabulary words to choose from, which decreases the quality of the words and increases the diversity of the words

(B) Increase the number of vocabulary words to choose from, which decreases the quality of the words and increases the diversity of the words

(C) Decreases the number of vocabulary words to choose from, which increases the quality of the words and increases the diversity of the words

(D) Decrease the number of vocabulary words to choose from, which increases the quality of the words and decreases the diversity of the words

Retrieval-Augmented Generation

Question Answering

- A subfield of NLP that focuses on developing models that can answer questions
- Generative LLMs have become very proficient at this
- In fact, many NLP tasks can be rephrased as a question and answer
- For example, classification:

  Review: "The movie was very exciting."

  What is the sentiment of the review: positive, negative, or neutral?

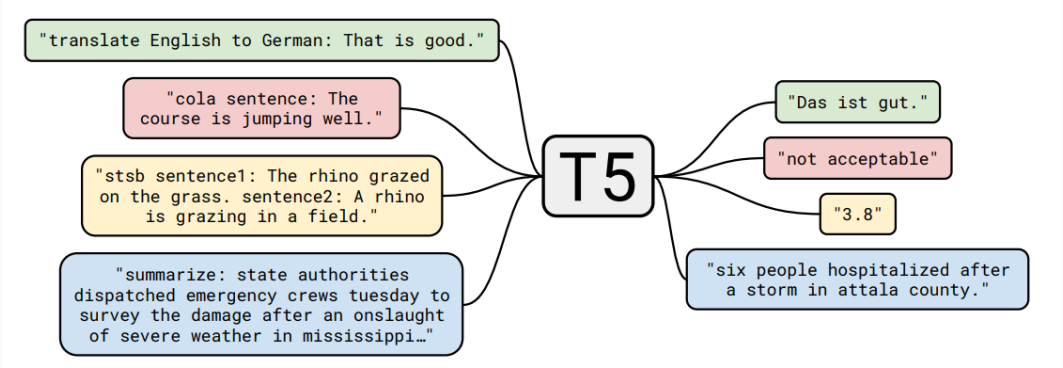


Figure from Raffel et al. (2020)

Generative LLMs Are Not Always Right

- LLMs often give the wrong answer—a problem known as **hallucination**
- LLMs can also be out of date. For example, the LLM might not know how to answer questions we may pose about the 2024 U.S. general election
- LLMs are also not well-calibrated
- A well-calibrated model is a model where the probability of generating the correct answer is highly correlated with the probability of the answer being correct
- But oftentimes, LLMs will give the wrong answer with high certainty

Two Potential Solutions

There are two potential solutions:

1. Include examples in the prompt and hope that the model learns from the examples
2. Use gradient-based updates (finetuning) to improve the model for a specific task

Few-Shot Learning

- Recall that in few-shot learning, we often include a few demonstrations that help the model learn about what we are trying to answer
- However, what may appear as good demonstrations to us may not be very good demonstrations for a model
- LLM may also detect unintentional patterns in the examples, which can cause overfitting
- In an ideal world, we would have a set of examples that are bespoke for the question at hand

Let's say we were interested in answering questions about Formula 1 driver Lewis Hamilton. We could just directly prompt the generative LLM with the question.

Let's say we were interested in answering questions about Formula 1 driver Lewis Hamilton. We could just directly prompt the generative LLM with the question.

Sir Lewis Carl Davidson Hamilton (born 7 January 1985) is a British racing driver, currently competing in Formula One for Mercedes. Hamilton has won a joint-record seven Formula One World Drivers' Championship titles—tied with Michael Schumacher—and holds the records for most wins (105), pole positions (104), and podium finishes (201), among others.

Born and raised in Stevenage, Hertfordshire, Hamilton began his career in karting aged six, winning several national titles and attracting the attention of Ron Dennis, who signed him the McLaren Young Driver Programme in 1998. After winning the direct-drive Karting World Cup and European Championship in 2000, Hamilton progressed to junior formulae, where his successes included winning the Formula 3 Euro Series and the GP2 Series. He subsequently signed for McLaren in 2007, becoming the first black driver to compete in Formula One at the Australian Grand Prix. In his rookie season, Hamilton won four Grands Prix and set several records as he finished runner-up to Kimi Räikkönen by one point. Hamilton won his maiden title in 2008, making a title-deciding overtake on the last lap of the last race of the season to become the then-youngest World Drivers' Champion. Despite Red Bull dominance throughout his remaining four seasons at McLaren, he achieved multiple race wins in each.

Hamilton signed for Mercedes in 2013 to partner Nico Rosberg, ending his 15-year association with McLaren. Following his maiden victory with the team at the 2013 Hungarian Grand Prix, new engine regulations came into effect the following season, which saw Mercedes emerge as the dominant force in Formula One. From 2014 to 2016, Hamilton and Rosberg won 51 of 59 Grands Prix amidst their fierce rivalry—widely known as *The Silver War*—with Hamilton winning the former titles in 2014 and 2015, and Rosberg winning the latter. Following Rosberg's retirement, Hamilton twice

Sir
Lewis Hamilton
MBE HonFREng

Hamilton at the 2021 Abu Dhabi Grand Prix

Born	Lewis Carl Davidson Hamilton 7 January 1985 (age 39) Stevenage, Hertfordshire, England
Partner	Nicole Scherzinger (2007–2015)
Relatives	Nicolas Hamilton (half-brother)
Awards	Full list

Retrieval-Augmented Generation (RAG)

- A formal way to use **information retrieval** techniques to retrieve documents that are likely to have information that can help us answer questions
- The LLM can then leverage these documents to generate an answer given these documents

Information Retrieval

- Subfield of computer science that involves the retrieval of all media based on information needs
- A very well-known information retrieval website is `google.com`
- You type in a search query, and it retrieves many webpages relevant to your search

RAG as an IR Task

- Ad hoc retrieval: user poses a query to a retrieval system
- Document: unit of text the system indexes and retrieves
- Collection: the set of documents being used to satisfy user requests
- Term: a word in a collection but may also include phrases
- Query: a user's information needs expressed as a set of terms

RAG

- **RAG**: process of improving the output of a generative LLM by referencing a knowledge base outside its training data sources before generating a response
- *We* get to control the knowledge base outside the training data
- We could, for example, provide a database of updated news articles to answer questions about current events

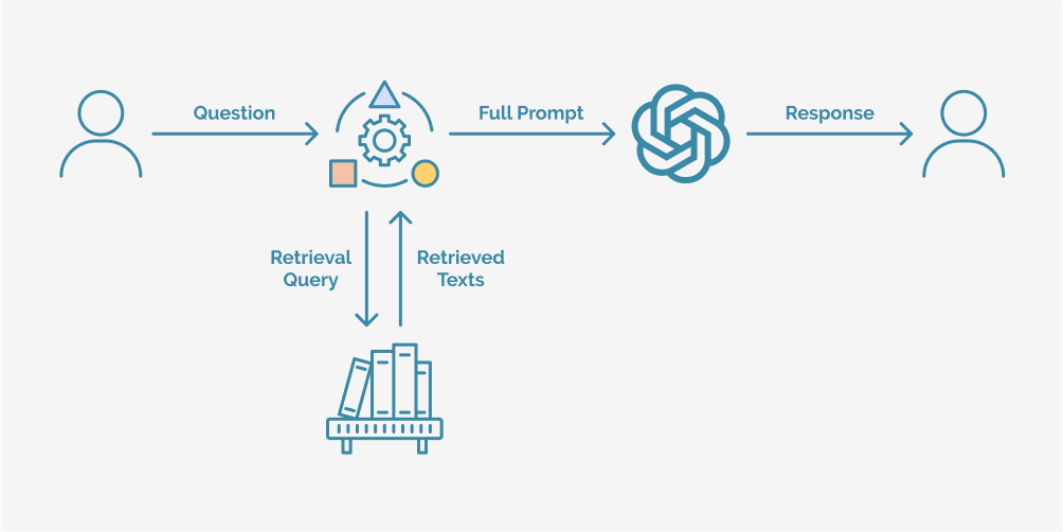


Diagram from Trantor

Document Embeddings

- We need a way to know which documents are most relevant to a particular prompt
- For example, we need to figure out a way of picking out the Wikipedia article for Lewis Hamilton if the prompt is asking a question about Lewis Hamilton
- We'll do this by converting our documents and our query into **document embeddings**
- In other words, we want to map each document to a vector

SentenceBERT

- An efficient way to do this would be to use SentenceBERT
- **SentenceBERT (SBERT)**: A modification of BERT that is optimized to generate meaningful sentence embeddings instead of just word-level embeddings
- We won't go into the details, but SBERT embeddings are suitable for comparison tasks (e.g., cosine similarity)
- For more information about SBERT, see https://sbert.net/
- As a note, document embedding and sentence embeddings are used interchangeably

Benefits of SBERT

- Generates high-quality embeddings for sentence similarity tasks
- Significantly faster than using BERT directly for sentence comparison tasks
- Useful for information retrieval tasks

Using SBERT with RAG

- We can embed our documents with SBERT
- We can embed each of our prompts (queries) with SBERT
- Then, choose the k documents that are closest to the query as measured by cosine similarity
- Include those documents as part of the full prompt
- Input the prompt to the generative LLM to answer

Advantages of RAG

- More computationally efficient: because we are not finetuning the model, this is often more computationally efficient and thus more cost-effective
- Allows you to update LLMs quickly: it might take a long time to update an LLM with the latest information, which is why companies such as OpenAI and Anthropic do not often update the cutoff date of their models
- Potentially more interpretable: because the model is pulling information from documents that *we* control, this can potentially make the solutions generated by an LLM more interpretable

Disadvantages of RAG

- Complexity: RAG involves a retriever aspect, which can lead to greater model complexity
- Dependency on the Retriever Quality: in our case, we're dependent on SBERT actually being a good way to calculate sentence embeddings as a way to get us good documents
- Does not make up for a model that struggles in pattern recognition: if a model struggles to recognize patterns in whatever application we're interested in, this can't be solved using RAG

Lab

We'll now turn the lab, which can be found here.