

Lecture 7: Prompting

[Giới thiệu](#)

[Prompting Fundamentals](#)

[Basic Prompting \(Radford et al. 2018\)](#)

[Standard Prompting Workflow](#)

[Prompt Templates](#)

[Chat Prompts](#)

[Chat Prompts Behind The Scenes](#)

[Answer Prediction](#)

[Post-processing](#)

[Output Formatting](#)

[Output Selection](#)

[Output Mapping](#)

[Few-shot Prompting/ In-context Learning](#)

[Few-shot Prompting \(Brown+ 2020\)](#)

[Few-shot Prompting w/ Chat Prompts \(OpenAI Cookbook\)](#)

[LMs are Sensitive to Small Changes in In-Context Examples](#)

[But Effects are Sometimes Counter-intuitive \(Min et al. 2022\)](#)

[Chain-of-thought Prompting](#)

[Chain of Thought Prompting \(Wei et al. 2022\)](#)

[Unsupervised Chain-of-thought Prompting \(Kojima et al. 2022\)](#)

[Prompting and Programs](#)

[Structuring Outputs as Programs can Help \(Madaan et al. 2022\)](#)

[Program-aided Language Models \(Gao et al. 2022\)](#)

[Prompt Engineering](#)

[Design of Prompts](#)

[Manual Engineering: Format](#)

[Manual Engineering: Instructions](#)

[Methods for Automating Prompt Engineering](#)

[Prompt paraphrasing](#)

[Gradient-based Search \(Shin et al. 2020\)](#)

[Prompt Tuning \(Lester et al. 2021\)](#)

[Prefix Tuning \(Li and Liang 2021\)](#)

[Prompting and Fine Tuning](#)

[Connection to Other Training Methods](#)

[Prompting as a Prior](#)

[Resources](#)

Giới thiệu

Trong bài hôm nay, chúng ta sẽ khám phá về khái niệm "prompting". Đây là một phương pháp mới xuất hiện trong vài năm gần đây, trở thành tiêu chuẩn trong việc tương tác với các mô hình. Cụ thể, chúng ta khuyến khích một mô hình đã được huấn luyện trước đưa ra dự đoán bằng cách cung cấp một "textual prompt" (lời nhắc bằng văn bản) để chỉ định nhiệm vụ cần thực hiện. Đây chính là cách mà bạn thường tương tác với các công cụ như "chat GPT" hay những ứng dụng tương tự.

Prompting Fundamentals

Basic Prompting (Radford et al. 2018)

Cách thức hoạt động cơ bản của prompting là bạn sẽ thêm một chuỗi văn bản vào đầu vào của mô hình và hoàn thành nó. Cách hoàn thành này có thể dựa trên nhiều phương pháp sinh khác nhau như beam search, sampling, MBR, hay self-consistency.

Ví dụ, khi tôi nhập câu "when a dog sees a squirrel it will usually" vào mô hình GPT-2 nhỏ, kết quả trả về là "Be Afraid of Anything unusual as an exception that's when a squirrel is usually afraid to bite." Như bạn thấy, nếu mô hình không mạnh mẽ, phản hồi có thể không được tốt lắm.

Khi tôi thử nghiệm với GPT-2 XL, kết quả là "when a dog sees a squirrel it will usually lick the squirrel it will also touch its nose to the squirrel the tail and nose if it can." Điều này có thể đúng, nhưng cũng không phải là câu trả lời điển hình.

Tôi đã sử dụng phương pháp sampling thông thường với temperature là 1 mà không áp dụng top-k hay top-p. Nếu tôi thay đổi các tham số sinh, chẳng hạn như đặt top-k là 50 và top-p là 0.95, chúng ta có thể thấy những kết quả khác nhau.

Khi tôi thử nghiệm với các tham số mới, phản hồi là "when a dog sees a squirrel it will usually get angry, scratched the squirrel and run off.". Phản hồi này có vẻ thực tế hơn so với câu trả lời trước đó.

Điều này cho thấy rằng việc điều chỉnh các tham số sinh có thể tạo ra những phản hồi khác nhau và đôi khi hợp lý hơn.

Standard Prompting Workflow

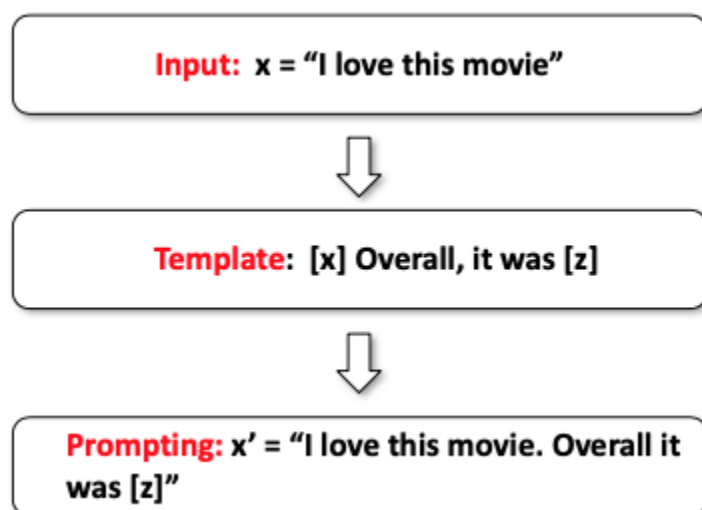
Việc sử dụng prompting là một phương pháp cơ bản để giải quyết các vấn đề. Mặc dù việc hoàn thiện văn bản là một ứng dụng quan trọng và hữu ích, như việc tôi thường xuyên sử dụng tính năng hoàn thiện văn bản trong Gmail với các gợi ý về những gì tôi nên viết tiếp, nhưng prompting không chỉ dừng lại ở đó.

Chẳng hạn, khi bạn sử dụng tính năng tab autocomplete trên điện thoại, bạn cũng đang áp dụng một mô hình ngôn ngữ. Tuy nhiên, rất thường xuyên, chúng ta sử dụng prompting để thực hiện những nhiệm vụ khác ngoài việc hoàn thiện văn bản.

Quy trình tiêu chuẩn để giải quyết các nhiệm vụ NLP bằng cách sử dụng prompting thường bao gồm ba bước: đầu tiên, chúng ta điền vào một mẫu prompt; tiếp theo, dự đoán câu trả lời; và cuối cùng, xử lý kết quả dự đoán theo một cách nào đó.

Prompt Templates

Các mẫu prompt (prompt templates) là những cấu trúc mà chúng ta sẽ điền vào bằng các đầu vào cụ thể. Ví dụ, nếu chúng ta có một đầu vào X, chẳng hạn như "I love this movie", thì mẫu của chúng ta có thể là "X overall it was Z". Khi chúng ta muốn thực hiện một dự đoán, chúng ta sẽ chuyển đổi điều này thành một prompt thực tế để đưa vào mô hình ngôn ngữ bằng cách điền vào mẫu. Cụ thể, chúng ta sẽ có "I love this movie overall it was blank" và sau đó điền vào phần tiếp theo.



Chat Prompts

Trong thời đại hiện nay, chúng ta thường sử dụng một loại mô hình rất phổ biến, đặc biệt là các chatbot. Thực tế, ngay cả khi các mô hình không được huấn luyện như chatbot, chúng vẫn có thể hoạt động hiệu quả trong một số trường hợp nhất định. Một trong những cách để tương tác với các mô hình này là thông qua "chat prompt".

Thông thường, chúng ta sẽ chỉ định các đầu vào theo định dạng gọi là "open AI messages format". Định dạng này bao gồm một danh sách các đầu ra, trong đó mỗi mục được gán một vai trò và nội dung cụ thể. Ví dụ:

```
messages=[ {  
    "role": "system",  
    "content": "Please classify movie reviews as 'positive' or 'negative'."
```

```

},
{
  "role": "user",
  "content": "This movie is a banger."
}, ]

```

Các vai trò trong định dạng này bao gồm:

- “system”: là thông điệp được cung cấp cho hệ thống nhằm ảnh hưởng đến hành vi của nó.
- “user”: là thông điệp đầu vào từ người dùng.
- “assistant”: là thông điệp đầu ra từ hệ thống.

Vai trò "system" giúp giải thích cách mà hệ thống nên hoạt động, cho phép nó hiểu rõ hơn về nhiệm vụ của mình. Ngoài ra, nếu bạn có một cuộc hội thoại nhiều lượt, định dạng này cũng có thể bao gồm các vai trò "user" và "assistant" luân phiên, ví dụ như "user" rồi đến "assistant", và tiếp tục như vậy. Điều này giúp làm rõ rằng đây là một cuộc đối thoại nhiều lượt.

Chat Prompts Behind The Scenes

Khi chúng ta sử dụng các prompt trong chat, những prompt này sẽ được chuyển đổi thành chuỗi token và sau đó được đưa vào mô hình. Mặc dù cách thức nhập liệu có vẻ đặc biệt, thực tế là chúng chỉ đơn giản được đưa vào mô hình như một prompt. Những prompt này thực chất là các phiên bản đặc biệt của một mẫu (template). Ví dụ:

LLaMa	Alpaca
<pre> Sys. [INST] <<SYS>> You are an assistant that ... <</SYS>> [/INST] User [INST]This movie is great.[/INST] Asst. Positive. </pre>	<pre> Sys. ### Instruction: You are an assistant that ... User ### Instruction: This movie is great. Asst. ### Response: Positive. </pre>

Một công cụ hữu ích mà tôi muốn giới thiệu là Light LLM. Công cụ này giúp bạn dễ dàng truy vấn nhiều loại mô hình ngôn ngữ khác nhau, từ OpenAI đến các mô hình mã nguồn mở. Khi bạn sử dụng Light LLM, nó sẽ tự động chuyển đổi định dạng thông điệp của OpenAI thành định dạng prompt phù hợp cho mô hình mà bạn đang sử dụng.

Khi bạn bắt đầu với một input như "My account is a great movie" và đưa vào mô hình, kết quả sẽ phụ thuộc vào mô hình mà bạn đang sử dụng. Hệ thống thông điệp được đưa vào ban đầu không phải là một tính năng của các mô hình OpenAI, mà là một giải pháp để ngăn chặn các cuộc tấn công như "prompt injection" hay "jailbreaking". OpenAI đã huấn luyện các mô hình của

họ để không xuất ra những thông tin có trong thông điệp hệ thống, nhằm bảo vệ thông tin nhạy cảm.

Thông thường, các mẫu này được viết thủ công. Bạn có thể tạo ra một chuỗi Python lớn với nội dung như "Please classify the topic" và sau đó thay thế vào vị trí cần thiết. Khoảng 90-95% thời gian, người dùng sẽ tự viết các mẫu này khi quyết định nhiệm vụ mà họ muốn thực hiện, chẳng hạn như phân loại cảm xúc.

Cuối cùng, mặc dù chúng ta không biết chính xác định dạng của OpenAI, nhưng có thể giả định rằng nó tương tự như những gì đã được mô tả ở trên.

Answer Prediction

Tiếp theo, dựa trên một prompt nhất định, chúng ta sẽ sử dụng các thuật toán phù hợp để dự đoán câu trả lời. Ví dụ, với prompt "I love this movie. Overall it was [z]" chúng ta có thể dự đoán từ "fantastic". Tuy nhiên, kết quả dự đoán không nhất thiết phải là "fantastic"; nó có thể là một câu khác như "overall it was um a really fantastic movie that I liked a lot" hoặc một câu tương tự. Điều này cho thấy rằng hệ thống có khả năng tạo ra nhiều phản hồi khác nhau dựa trên ngữ cảnh và thông tin đã cho.

Post-processing

Trong quá trình xử lý đầu ra từ các mô hình ngôn ngữ, chúng ta cần thực hiện bước gọi là "postprocessing" để chọn lọc và định hình lại kết quả. Ví dụ, khi bạn tương tác với ChatGPT hoặc một mô hình trò chuyện khác, bạn có thể xem xét đầu ra dưới dạng văn bản nguyên bản. Tuy nhiên, có thể bạn sẽ cần định dạng lại đầu ra để dễ dàng trực quan hóa, chỉ chọn những phần mà bạn muốn sử dụng, hoặc ánh xạ đầu ra đó đến các hành động khác. Những bước này giúp tối ưu hóa việc sử dụng thông tin từ các mô hình ngôn ngữ, đảm bảo rằng chúng ta nhận được kết quả phù hợp nhất với nhu cầu của mình.

Output Formatting

Ví dụ, khi tôi yêu cầu: "please write a table with the last five presidents and their birth dates", ChatGPT đã nhanh chóng cung cấp cho tôi một bảng thông tin về năm tổng thống Mỹ gần đây nhất cùng với ngày sinh của họ. Cụ thể, bảng này bao gồm các tên: Joe Biden, Donald Trump, Barack Obama, George W. Bush và Bill Clinton.

Điều thú vị là bảng này được viết bằng định dạng markdown, cho phép nó được hiển thị một cách dễ nhìn. Việc này rất quan trọng khi xây dựng các hệ thống tương tác với người dùng, vì chúng ta cần có khả năng trình bày thông tin một cách trực quan. Tuy nhiên, cần lưu ý rằng một mô hình ngôn ngữ lớn chỉ có thể xuất ra văn bản, tức là một chuỗi các token.

Tiếp theo, tôi đã yêu cầu ChatGPT xuất thông tin này dưới định dạng JSON với câu lệnh: "output that in json format". ChatGPT đã đáp ứng bằng cách cung cấp thông tin dưới dạng

JSON, không chỉ đơn thuần là một chuỗi JSON lớn mà còn có cả định dạng màu sắc để dễ đọc hơn. Có lẽ, mô hình này đã sử dụng cú pháp như "triple hash" để định dạng.

Tuy nhiên, không phải lúc nào ChatGPT cũng hoàn hảo. Trong một số trường hợp, nó có thể mắc lỗi, như khi xuất ra một định dạng không chính xác ở cuối đoạn. Điều này cho thấy rằng ngay cả những mô hình tiên tiến nhất cũng có thể gặp phải sai sót.

Output Selection

Có một số phương pháp hữu ích mà bạn có thể áp dụng, đặc biệt khi không sử dụng trong các ứng dụng trực tiếp dành cho người dùng. Ví dụ: trích xuất thông tin hoặc đưa ra quyết định phân loại.

Khi phân tích văn bản, bạn thường chọn những thông tin có tính chất chỉ dẫn cho câu trả lời. Ví dụ, nếu ai đó nói "I love this movie overall it was a movie that was simply fantastic", bạn có thể trích xuất từ khóa "fantastic" để chỉ ra cảm xúc tích cực. Có nhiều phương pháp khác nhau để thực hiện việc này, và chúng cũng được sử dụng trong các tiêu chuẩn đánh giá mô hình ngôn ngữ.

Đối với các bài toán phân loại, bạn có thể xác định các từ khóa như "fantastic" để chỉ ra lớp phân loại. Một ví dụ khác là trong các bài toán hồi quy hoặc số học, bạn có thể xác định và trích xuất các con số để sử dụng làm câu trả lời. Đối với mã nguồn, bạn có thể trích xuất các đoạn mã và sử dụng cú pháp triple back ticks để thực thi mã đó.

Tất cả những phương pháp này đều là các phương pháp heuristic, nhưng chúng có thể giúp bạn lấy được câu trả lời mong muốn từ văn bản được tạo ra.

Output Mapping

Ánh xạ đầu ra là gì? Đó là, khi nhận được một câu trả lời, chúng ta cần ánh xạ nó vào một nhãn lớp hoặc giá trị liên tục. Ví dụ, từ "fantastic" có thể được ánh xạ vào lớp "positive". Nếu chúng ta muốn trích xuất đánh giá từ 1 đến 5 sao từ các bài đánh giá, việc này là rất cần thiết.

Thường thì, quá trình ánh xạ này có thể là một lớp đơn ánh xạ tới nhiều từ. Bằng cách thực hiện điều này, chúng ta có thể có được một ánh xạ vững chắc hơn tới con số mà chúng ta thực sự muốn. Gần đây, tôi đã thấy một ví dụ thú vị trên Twitter về việc này. Một người đã sử dụng GPT-4 để tạo ra một mô hình nhằm đánh giá các mô hình mã nguồn mở dựa trên phản hồi tốt và xấu. Họ bắt đầu với việc cho điểm từ 1 đến 5 sao, nhưng sau đó đã chuyển sang các mức đánh giá như "very good", "good", "okay", "bad", "very bad". Kết quả là mô hình GPT đã hoạt động tốt hơn rất nhiều với cách đánh giá này so với việc sử dụng thang điểm 1 đến 5.

Điều này cho thấy rằng chúng ta nên suy nghĩ nghiêm túc về cách ánh xạ đầu ra. Một cách để tiếp cận vấn đề này là xem xét khả năng xuất hiện của dữ liệu trong một tập hợp lớn các dữ liệu trên internet. Ví dụ, câu hỏi "How good is this movie?" với câu trả lời "very good" có thể có xác

suất xuất hiện cao hơn so với câu hỏi "How good is this movie?" với câu trả lời "5". Việc khai thác dữ liệu từ web có thể giúp chúng ta tìm ra những cách diễn đạt tốt nhất.

Khi nói về cách mà mô hình học hỏi, mô hình không chỉ dự đoán từ "fantastic" mà còn dự đoán mã token tương ứng, chẳng hạn như "73521". Nếu mô hình đã thấy mã token này xuất hiện thường xuyên hơn trong các bài đánh giá so với mã token cho số "1" hoặc "5", thì nó sẽ có khả năng cao hơn để dự đoán "fantastic" chính xác hơn là "five star". Điều này nhấn mạnh rằng chúng ta nên cân nhắc kỹ lưỡng về những gì mà mô hình đã thấy trong dữ liệu huấn luyện.

Một quy tắc quan trọng là không nên cố gắng làm cho một mô hình ngôn ngữ thực hiện những nhiệm vụ mà nó chưa từng thấy trong dữ liệu huấn luyện. Điều này sẽ giúp cho quá trình phát triển mô hình trở nên dễ dàng hơn rất nhiều.

Few-shot Prompting/ In-context Learning

Few-shot Prompting (Brown+ 2020)

Few-shot Prompting một kỹ thuật quan trọng mà chúng ta có thể áp dụng để cải thiện hiệu suất của các mô hình ngôn ngữ.

Few-shot prompting là phương pháp cung cấp một vài ví dụ về nhiệm vụ cùng với hướng dẫn. Ví dụ, bạn có thể viết một hướng dẫn như: "please classify movie reviews as positive or negative" và thêm vào các ví dụ như:

- Input: "I really don't like this movie" & Output: "negative"
- Input: "This movie is great" & Output: "positive"

Phương pháp này rất hiệu quả trong việc đảm bảo định dạng đầu ra đúng, đặc biệt là khi làm việc với các mô hình mạnh như GPT-4. Các mô hình này có khả năng tuân theo hướng dẫn tốt hơn. Ngược lại, với các mô hình yếu hơn, chúng có thể không tuân theo hướng dẫn một cách chính xác, ví dụ như:

- Input: "I really don't like this movie"
Output: "I think this is probably negative"

Cần lưu ý rằng "few-shot prompting" và In-context Learning (Học trong ngữ cảnh) thực chất là hai khái niệm tương tự, chỉ khác nhau về cách diễn đạt. "Few-shot" trái ngược với "zero-shot", trong đó "zero-shot" có nghĩa là không cung cấp ví dụ nào. Học trong ngữ cảnh có nghĩa là bạn đang học cách thực hiện một nhiệm vụ mà không cần cung cấp dữ liệu tinh chỉnh, mà thay vào đó là các ví dụ trong ngữ cảnh của mô hình ngôn ngữ.

Few-shot Prompting w/ Chat Prompts (OpenAI Cookbook)

Trong phần này, chúng ta sẽ thảo luận về cách sử dụng mô hình OpenAI một cách hiệu quả. Có một số cách để thực hiện điều này. Một phương pháp là thiết lập vai trò là "user" và "assistant",

sau đó thêm lịch sử hội thoại vào các tin nhắn gửi đến mô hình ngôn ngữ. Tuy nhiên, cách được khuyến nghị trong tài liệu hướng dẫn của OpenAI là gửi thông tin này dưới dạng một "system message" và cung cấp thêm biến tên như "example user" và "example assistant".

```

{
  "role": "system",
  "name": "example_user",
  "content": "New synergies will help drive top-line growth."
},
{
  "role": "system",
  "name": "example_assistant",
  "content": "Things working well together will increase revenue."
},

```

Lý do chính cho việc này là nếu bạn không làm như vậy, mô hình có thể hiểu các ví dụ trong few shot như là những gì đã xảy ra trước đó trong cuộc hội thoại. Ngược lại, khi gửi dưới dạng "system message", bạn có thể đảm bảo rằng mô hình sẽ không tham chiếu lại những ví dụ đó, điều này giúp bảo vệ quyền riêng tư hơn là vấn đề chính xác.

Tuy nhiên, nếu bạn đang sử dụng mô hình mã nguồn mở, bạn cần lưu ý rằng tên này có thể không được bao gồm trong mẫu prompt. Ví dụ, trong các mẫu prompt của "light llm" mà tôi đã gửi, tên này thậm chí không được đề cập, dẫn đến việc bạn có thể nhận được một thông điệp hệ thống không rõ ràng. Do đó, hãy cẩn thận với điều này.

LMs are Sensitive to Small Changes in In-Context Examples

- Example ordering (Lu et al. 2021)
- Label balance (Zhang et al. 2022)

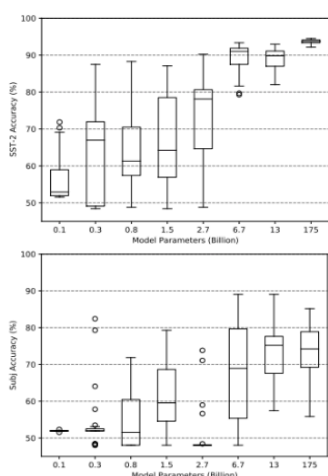
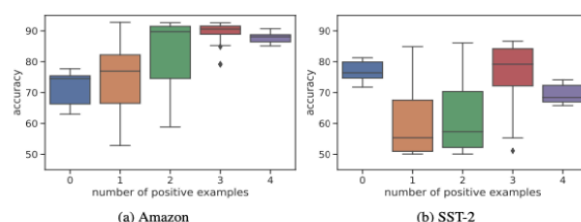
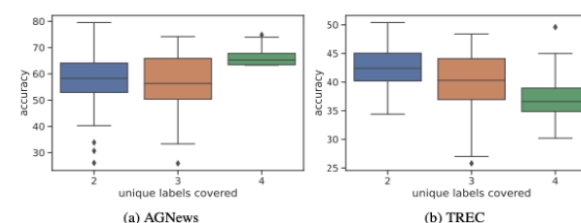


Figure 1: Four-shot performance for 24 different sample orders across different sizes of GPT-family models (GPT-2 and GPT-3) for the SST-2 and Subj datasets.



- Label coverage (Zhang et al. 2022)



Các mô hình ngôn ngữ lớn (LLMs) rất nhạy cảm với những thay đổi nhỏ trong ngữ cảnh và ví dụ mà bạn cung cấp. Một nghiên cứu của Luol đã chỉ ra rằng thứ tự của các ví dụ có thể ảnh hưởng lớn đến kết quả. Cụ thể, nếu bạn sắp xếp lại cùng một tập ví dụ theo thứ tự khác nhau,

kết quả có thể thay đổi rất nhiều, đặc biệt là với các mô hình nhỏ như GPT-2, trong khi các mô hình lớn hơn như GPT-3.5 ít bị ảnh hưởng hơn.

Ngoài ra, sự cân bằng nhãn cũng rất quan trọng. Ví dụ, trong bài toán phân loại cảm xúc, nếu bạn chỉ có các ví dụ tích cực hoặc tiêu cực, điều này có thể ảnh hưởng đến độ chính xác. Trong bộ dữ liệu đánh giá Amazon, hầu hết các đánh giá đều tích cực, vì vậy việc có nhiều ví dụ tích cực sẽ giúp cải thiện độ chính xác. Ngược lại, trong bộ dữ liệu SST-2, việc có sự cân bằng giữa các nhãn sẽ cho kết quả tốt hơn so với việc chỉ có một loại nhãn.

Một yếu tố khác cần xem xét là độ bao phủ nhãn trong phân loại đa lớp. Việc đảm bảo có đủ đại diện cho tất cả các lớp mong muốn là rất quan trọng, nhưng nếu có quá nhiều nhãn thiếu sót, điều này có thể gây nhầm lẫn cho mô hình.

Cuối cùng, khi sử dụng GPT để đánh giá dịch máy, việc thêm các đầu ra có điểm số cao, thấp và trung bình là rất quan trọng. Điều này cũng áp dụng cho các bài toán hồi quy.

But Effects are Sometimes Counter-intuitive (Min et al. 2022)

Việc áp dụng các ví dụ trong ngữ cảnh (in-context examples) để cải thiện độ chính xác của mô hình là một chủ đề thú vị nhưng cũng đầy thách thức. Hiện tại, chưa có quy tắc cụ thể nào hướng dẫn cách thức xây dựng các ví dụ này một cách hiệu quả.

Một nghiên cứu đáng chú ý đã chỉ ra rằng việc sử dụng các ví dụ trong ngữ cảnh có thể mang lại kết quả tốt hơn, ngay cả khi các nhãn (labels) của ví dụ đó không chính xác. Cụ thể, nghiên cứu này cho thấy rằng khi họ sử dụng các ví dụ trong ngữ cảnh nhưng ngẫu nhiên thay đổi nhãn, thậm chí khi nhãn hoàn toàn sai, mô hình vẫn đạt được độ chính xác cao hơn so với việc không sử dụng bất kỳ ví dụ nào. Điều này cho thấy rằng mô hình có thể đang học cách định dạng và nhận diện các nhãn, thay vì chỉ đơn thuần học từ dữ liệu huấn luyện.

Khi áp dụng các ví dụ trong ngữ cảnh, chúng ta không huấn luyện lại mô hình mà chỉ cung cấp cho nó một số ví dụ để nó có thể hoàn thành các nhiệm vụ tiếp theo. Điều này thật sự ấn tượng, đặc biệt là với những mô hình chưa được huấn luyện một cách cụ thể cho nhiệm vụ đó. Một lý do cho sự thành công này có thể là do sự lặp lại của các mẫu văn bản trên internet, nơi mà các bài toán thường được trình bày theo dạng "question one - math problem - answer".

Tuy nhiên, cần lưu ý rằng việc cung cấp quá nhiều ví dụ có thể làm giảm độ chính xác của mô hình, đặc biệt trong các bài toán phân loại nhị phân. Khi có nhiều ví dụ hơn, mô hình có thể bị rối và quên đi các hướng dẫn quan trọng do ngữ cảnh trở nên dài hơn.

Cuối cùng, một câu hỏi thú vị được đặt ra là liệu việc các ví dụ trong ngữ cảnh phản ánh tốt phân phối dữ liệu có giúp tăng độ chính xác hay không. Câu trả lời có thể là có, nhưng điều quan trọng hơn có thể là việc bao phủ tốt hơn các nhãn, ngay cả khi có một số nhãn thiếu sót. Điều này đặc biệt đúng với các mô hình mạnh hơn.

Chain-of-thought Prompting

Chain of Thought Prompting (Wei et al. 2022)

Phương pháp Chain of Thought prompting yêu cầu mô hình giải thích quá trình suy luận của nó trước khi đưa ra câu trả lời.

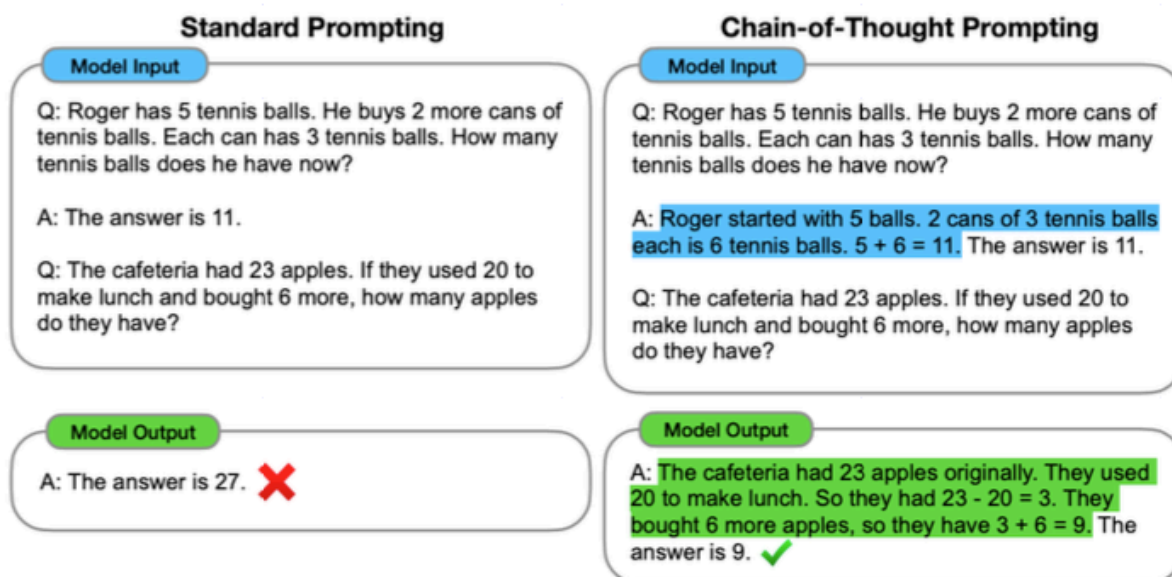


Figure 1: Chain-of-thought prompting enables large language models to tackle complex arithmetic, commonsense, and symbolic reasoning tasks. Chain-of-thought reasoning processes are highlighted.

Ví dụ bên trái là cách mà mô hình có thể được kích thích để suy nghĩ trước khi đưa ra câu trả lời. Với Ví dụ bên phải, nếu không có quá trình suy luận, mô hình có thể đưa ra câu trả lời sai.

Chain of Thought prompting không chỉ đơn thuần là đưa ra câu trả lời mà còn cung cấp một chuỗi lý luận bổ sung. Khi được cung cấp chuỗi lý luận này, mô hình sẽ có khả năng cao hơn để đưa ra câu trả lời chính xác.

Có hai lý do chính giải thích tại sao phương pháp này hiệu quả. Thứ nhất, nó cho phép mô hình phân tích các vấn đề phức tạp thành các vấn đề đơn giản hơn, từ đó dễ dàng hơn để giải quyết. Thay vì cố gắng giải quyết toàn bộ vấn đề ngay lập tức, mô hình sẽ giải quyết từng phần một, ví dụ như tính số lượng còn lại sau khi mua, rồi sau đó cộng thêm số lượng mới.

Lý do thứ hai là nó cho phép thời gian tính toán thích ứng. Trong một mô hình Transformer, thời gian tính toán cho việc dự đoán mỗi token là cố định. Một số vấn đề khó hơn những vấn đề khác, vì vậy việc sử dụng một mô hình lớn để giải quyết các bài toán phức tạp trong cùng một khoảng thời gian với việc dự đoán một từ đơn giản là không hiệu quả. Chain of Thought reasoning giúp mô hình có thêm thời gian để giải quyết các vấn đề khó hơn.

Unsupervised Chain-of-thought Prompting (Kojima et al. 2022)

Trong phần này, chúng ta sẽ khám phá một nghiên cứu thú vị liên quan đến khả năng suy luận của các mô hình ngôn ngữ. Câu hỏi đặt ra là: "Điều gì sẽ xảy ra nếu chúng ta yêu cầu mô hình suy luận?" Câu trả lời là: "Nó vẫn hoạt động hiệu quả."

Bài báo so sánh giữa hai phương pháp: "few shot learning" và "few shot Chain of Thought", trong đó bạn cung cấp các ví dụ về chuỗi suy luận, và "zero shot prompting" cùng với "zero shot Chain of Thought". Cụ thể, họ chỉ cần thêm cụm từ "let's think step by step" vào cuối câu lệnh, điều này kích thích mô hình thực hiện suy luận theo chuỗi mà không cần thêm ví dụ nào về cách thức hoạt động của chuỗi suy luận đó.

(a) Few-shot	(b) Few-shot-CoT
<p>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?</p> <p>A: The answer is 11.</p> <p>Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?</p> <p>A:</p> <p>(Output) The answer is 8. ✗</p>	<p>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?</p> <p>A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.</p> <p>Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?</p> <p>A:</p> <p>(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are $16 / 2 = 8$ golf balls. Half of the golf balls are blue. So there are $8 / 2 = 4$ blue golf balls. The answer is 4. ✓</p>
(c) Zero-shot	(d) Zero-shot-CoT (Ours)
<p>Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?</p> <p>A: The answer (arabic numerals) is</p> <p>(Output) 8 ✗</p>	<p>Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?</p> <p>A: Let's think step by step.</p> <p>(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓</p>

Tại sao điều này lại hiệu quả? Bởi vì trên internet có rất nhiều ví dụ về giải quyết bài toán toán học hoặc tập dữ liệu QA, nơi mà cụm từ "let's think step by step" thường được sử dụng. Điều này cho thấy rằng đây là một trực giác tốt mà bài báo đã trả lời và thực sự hiệu quả.

Một điều thú vị là khi tôi thử nghiệm với Chat GPT và đưa ra một bài toán: "I am teaching a class with 98 students. 70% turn in the assignment on time. 10% hand it in late. How many did not turn in?" Mô hình đã tự động viết mã cho tôi, điều này là một tính năng mà tôi không mong muốn. Mặc dù tôi không yêu cầu mô hình suy luận theo chuỗi, nhưng lý do là vì Chat GPT đã được tinh chỉnh để thực hiện suy luận này, ngay cả khi không có yêu cầu cụ thể.

Do đó, nếu bạn muốn tìm ra cách tốt hơn để kích thích điều này từ một mô hình thô, bạn sẽ cần sử dụng một mô hình thô như Llama 2 mà không có tinh chỉnh chat, để có thể thực hiện trong một môi trường trung lập, không bị ảnh hưởng bởi dữ liệu huấn luyện trước đó.

Prompting and Programs

Structuring Outputs as Programs can Help (Madaan et al. 2022)

Nghiên cứu gần đây cho thấy việc cấu trúc đầu ra dưới dạng chương trình có thể cải thiện kết quả, ngay cả khi nhiệm vụ không liên quan trực tiếp đến lập trình. Trong một thí nghiệm về dự đoán kiến thức thủ tục (ví dụ như cách nấu bánh), các nhà nghiên cứu so sánh hiệu quả của ba định dạng đầu ra: văn bản thuần túy, định dạng DOT để vẽ đồ thị, và mã Python.

Kết quả cho thấy cấu trúc đầu ra dưới dạng mã Python hiệu quả nhất. Lý do chính là vì mô hình ngôn ngữ đã được đào tạo với rất nhiều mã Python, giúp nó dự đoán chính xác hơn. Hơn nữa, cấu trúc chặt chẽ của mã khuyến khích mô hình tham chiếu đến các phần tử đã định nghĩa trước đó, giảm thiểu việc sinh ra nội dung ngẫu nhiên hoặc phi logic.

Một mẹo hữu ích khác là sử dụng định dạng JSON cho đầu ra có cấu trúc. Mô hình thường tuân thủ định dạng JSON chính xác hơn so với các hướng dẫn định dạng thông thường, giúp dễ dàng trích xuất thông tin mong muốn mà không cần viết bộ phân tích cú pháp phức tạp.

Tóm lại, việc cấu trúc đầu ra dưới dạng chương trình, đặc biệt là Python hoặc JSON, có thể cải thiện đáng kể chất lượng và tính nhất quán của kết quả từ các mô hình ngôn ngữ lớn, ngay cả đối với các tác vụ không liên quan trực tiếp đến lập trình.

Program-aided Language Models (Gao et al. 2022)

Trong một nghiên cứu gần đây về mô hình ngôn ngữ, các tác giả đã phát triển một phương pháp gọi là "program aided language models". Phương pháp này sử dụng chương trình để tạo ra đầu ra, cho phép độ chính xác cao hơn so với việc yêu cầu một mô hình ngôn ngữ (LM) thực hiện nhiệm vụ. Thay vì sử dụng phương pháp "Chain of Thought prompting", các tác giả đã tạo ra một số ví dụ Few-shot, trong đó có văn bản tiếng Anh và mã tương ứng. Cuối cùng là thực hiện mã để nhận được kết quả.

Hiện nay, phương pháp này đã được triển khai trong Chat GPT, cho phép người dùng viết yêu cầu và mô hình sẽ quyết định xem có nên tạo mã hay văn bản tùy thuộc vào loại vấn đề. Điều này đặc biệt hữu ích cho các câu hỏi liên quan đến số liệu, như việc tính toán thuế. Chat GPT cũng có khả năng thực hiện các trực quan hóa dữ liệu, ví dụ như khi tôi cung cấp một danh sách Python lớn và yêu cầu nó tạo ra một histogram, nó đã làm rất tốt và cung cấp cả mã để tôi có thể chỉnh sửa sau này.

Việc sử dụng Chat GPT không nên thay thế hoàn toàn việc học tập và làm việc, vì nó có thể mắc lỗi tinh vi. Cuối cùng, phương pháp này có thể được mở rộng thành các agents và tools, và nó sẽ được trình bày trong các bài viết sau.

Chain-of-Thought (Wei et al., 2022)

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 tennis balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: The bakers at the Beverly Hills Bakery baked 200 loaves of bread on Monday morning. They sold 93 loaves in the morning and 39 loaves in the afternoon. A grocery store returned 6 unsold loaves. How many loaves of bread did they have left?

Model Output

A: The bakers started with 200 loaves. They sold 93 in the morning and 39 in the afternoon. So they sold $93 + 39 = 132$ loaves. The grocery store returned 6 loaves. So they had $200 - 132 - 6 = 62$ loaves left.
The answer is 62.



Program-aided Language models (this work)

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 tennis balls.
`tennis_balls = 5`
2 cans of 3 tennis balls each is
`bought_balls = 2 * 3`
tennis balls. The answer is
`answer = tennis_balls + bought_balls`

Q: The bakers at the Beverly Hills Bakery baked 200 loaves of bread on Monday morning. They sold 93 loaves in the morning and 39 loaves in the afternoon. A grocery store returned 6 unsold loaves. How many loaves of bread did they have left?

Model Output

A: The bakers started with 200 loaves
`loaves_baked = 200`
They sold 93 in the morning and 39 in the afternoon
`loaves_sold_morning = 93`
`loaves_sold_afternoon = 39`
The grocery store returned 6 loaves.
`loaves_returned = 6`
The answer is
`answer = loaves_baked - loaves_sold_morning`
`- loaves_sold_afternoon + loaves_returned`

```
>>> print(answer)
74
```



Prompt Engineering

Design of Prompts

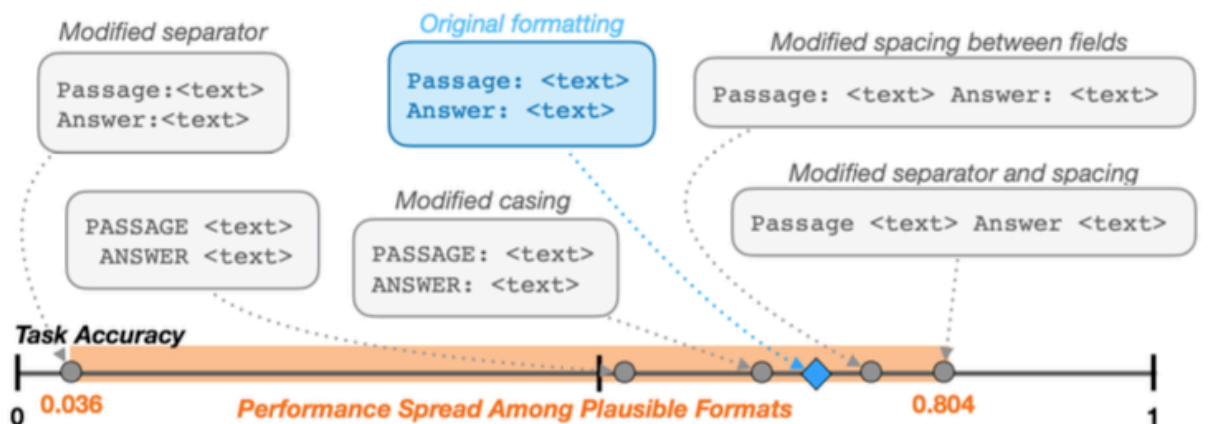
Trong kỹ thuật prompt, có nhiều cách để thiết kế các prompt hiệu quả. Một trong những phương pháp là thực hiện thủ công, nơi bạn có thể cấu hình một mẫu dựa trên các đặc điểm của nhiệm vụ, sử dụng tất cả những kiến thức đã được đề cập trước đó.

Ngoài ra, bạn cũng có thể thực hiện tìm kiếm tự động cho các prompt. Có một số phương pháp khác nhau để thực hiện tìm kiếm tự động này. Phương pháp đầu tiên là tìm kiếm trong không gian rời rạc (discrete space), nơi bạn tìm một prompt mà về cơ bản là văn bản. Phương pháp thứ hai là tìm kiếm trong không gian liên tục (continuous space), trong đó bạn tìm một prompt không phải là văn bản, nhưng vẫn là một prompt tốt.

Manual Engineering: Format

Việc đảm bảo định dạng đầu vào (prompt) phù hợp với mô hình đã được huấn luyện là rất quan trọng vì nó có thể ảnh hưởng lớn đến hiệu suất của mô hình. Một nghiên cứu đã chỉ ra rằng

những thay đổi nhỏ trong định dạng có thể dẫn đến sự khác biệt đáng kể trong độ chính xác của mô hình.



Ví dụ, nếu bạn thay đổi khoảng cách giữa các trường thông tin trong một prompt, điều này có thể làm tăng điểm số của bạn lên vài phần trăm. Ngược lại, việc loại bỏ dấu hai chấm (colons) cũng có thể cải thiện điểm số. Tuy nhiên, nếu bạn thay đổi cách viết chữ (casing) hoặc quên thêm khoảng trắng giữa đoạn văn và văn bản, điều này có thể làm giảm độ chính xác một cách nghiêm trọng. Cụ thể, việc quên thêm khoảng trắng có thể khiến độ chính xác giảm xuống chỉ còn 0.036%, trong khi chỉ cần thêm khoảng trắng có thể nâng cao độ chính xác lên tới 75%.

Một điểm thú vị là khi xem xét các mô hình khác nhau, có thể thấy rằng nhiều định dạng khả thi khác nhau thường cho kết quả độ chính xác thấp. Tuy nhiên, vẫn có một số định dạng ngoại lệ cho kết quả rất tốt, có thể tương ứng với những gì mô hình đã được huấn luyện. Do đó, điều quan trọng là bạn phải sử dụng định dạng prompt chuẩn cho mô hình mà bạn đang làm việc.

Manual Engineering: Instructions

Khi giao tiếp với các mô hình ngôn ngữ, việc đưa ra hướng dẫn rõ ràng, ngắn gọn và dễ hiểu là rất quan trọng. Một điều thú vị là việc tạo prompt cho các mô hình ngôn ngữ hiện nay, đặc biệt là với GPT-4, khá giống với việc giao tiếp với con người. Nếu bạn không giỏi trong việc giải thích cho người khác, điều này có thể gây khó khăn khi làm việc với mô hình. Thực hành giải thích cho các mô hình có thể là một cách tốt để cải thiện kỹ năng giao tiếp của bạn mà không làm phiền bạn bè.

Để bắt đầu, bạn nên sử dụng các hướng dẫn đơn giản. Hãy cho mô hình biết chính xác nó cần làm gì, ví dụ như "write", "classify", "summarize", "translate", "order". Một ví dụ về độ chính xác trong hướng dẫn là: "Explain the concept of prompt engineering. Keep the explanation short, only a few sentences and don't be too descriptive." Câu này yêu cầu mô hình giải thích khái niệm một cách ngắn gọn và không quá chi tiết.

Một điểm thú vị khác là khi bạn yêu cầu bạn bè làm điều gì đó mà họ không biết cách thực hiện, họ sẽ phân nản với bạn. Tuy nhiên, hiện tại, các mô hình ngôn ngữ không có khả năng phân

này. Do đó, khi giao tiếp với mô hình, bạn cần phải chính xác, vì chúng không cung cấp phản hồi khi bạn không rõ ràng trong yêu cầu của mình.

Tips: <https://www.promptingguide.ai/introduction/tips>

Methods for Automating Prompt Engineering

- Prompt paraphrasing
- Gradient-based discrete prompt search
- Prompt tuning
- Prefix tuning

Prompt paraphrasing

Một phương pháp thú vị là sử dụng paraphrasing để tạo ra các biến thể của một prompt hiện có. Phương pháp này khá đơn giản: bạn chỉ cần đưa một prompt vào một mô hình paraphrasing, và nó sẽ trả về những prompt mới. Điều này rất hữu ích vì các prompt được tạo ra thường mang tính tự nhiên hơn.

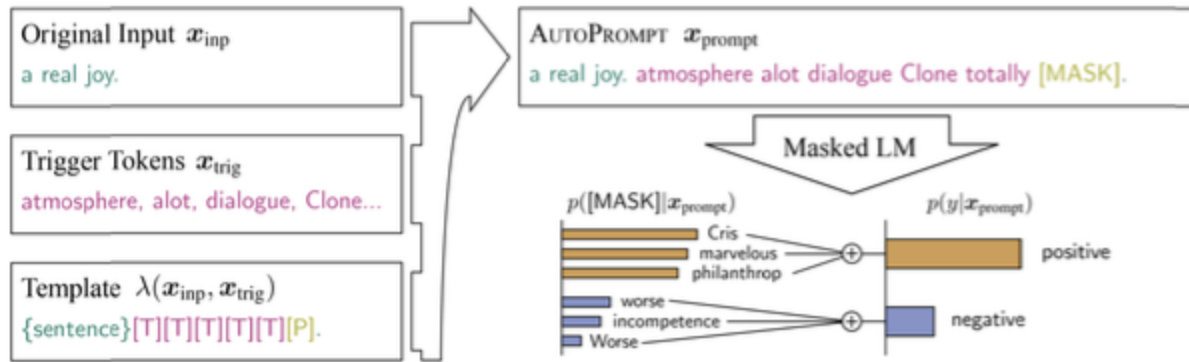
Bạn có thể thực hiện quá trình paraphrase này tới 50 lần, thử nghiệm với tất cả các biến thể và xem biến thể nào mang lại độ chính xác cao nhất, sau đó sử dụng biến thể đó.

Ngoài ra, có một nghiên cứu thú vị cho thấy bạn có thể thực hiện quá trình này một cách lặp đi lặp lại (Zhou et al. 2021). Cụ thể, bạn có thể paraphrase một lần, sau đó lọc ra những biến thể hoạt động tốt, và tiếp tục paraphrase chúng. Quá trình này có thể được lặp lại nhiều lần và thường mang lại kết quả tốt hơn so với việc chỉ thực hiện một lần duy nhất.

Gradient-based Search (Shin et al. 2020)

Một phương pháp tiên tiến trong việc tối ưu hóa prompt cho các mô hình ngôn ngữ lớn (LLMs) là tìm kiếm dựa trên gradient. Phương pháp này yêu cầu một mô hình có khả năng tính toán gradient và hoạt động như sau:

1. Tạo một prompt ban đầu.
2. Xem mỗi token trong prompt (T1, T2, T3, T4, T5) như các embedding riêng biệt.
3. Thực hiện lan truyền ngược (backpropagation) vào các embedding này.
4. Tối ưu hóa các embedding để đạt độ chính xác cao trên tập dữ liệu mục tiêu.
5. Sau khi tối ưu hóa, gán mỗi embedding tới embedding gần nhất trong không gian từ vựng của mô hình.



Ví dụ, kết quả cuối cùng có thể là một chuỗi như "atmosphere a lot dialogue clone totally". Mặc dù prompt này có vẻ không tự nhiên, nó có thể mang lại hiệu quả tốt hơn so với việc diễn đạt lại prompt bằng ngôn ngữ tự nhiên.

Phương pháp này đặc biệt phổ biến trong lĩnh vực tấn công đối kháng (adversarial attacks) trên các mô hình ngôn ngữ. Một nghiên cứu đáng chú ý là "Universal and transferable adversarial attacks on aligned language models" từ CMU, trong đó các nhà nghiên cứu tối ưu hóa prompt để khiến mô hình thực hiện các hành vi không mong muốn. Kết quả nghiên cứu cho thấy:

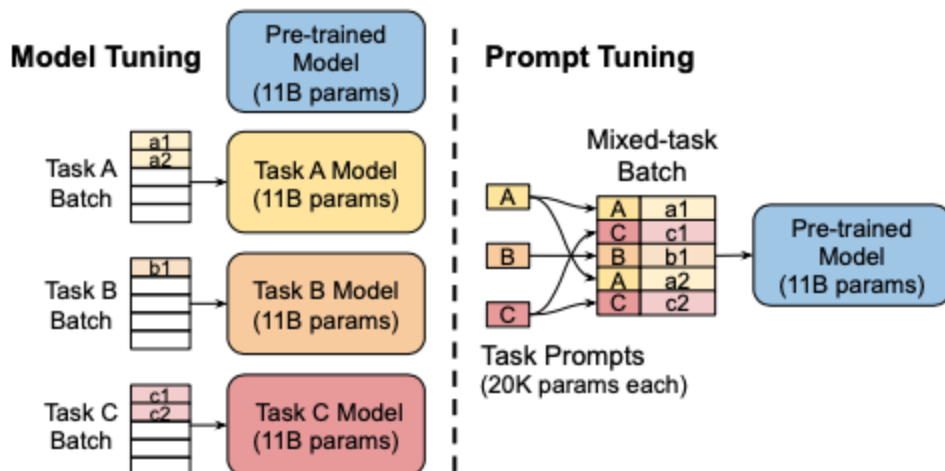
1. Có thể tạo ra các prompt khiến các mô hình như LLaMA tạo ra nội dung độc hại hoặc nguy hiểm.
2. Các prompt này thường có hiệu quả trên nhiều mô hình khác nhau, kể cả các mô hình của OpenAI như GPT.

Điều này gợi ý rằng các lỗ hổng này không chỉ đặc thù cho một mô hình cụ thể mà có thể là đặc điểm chung của các LLM.

Prompt Tuning (Lester et al. 2021)

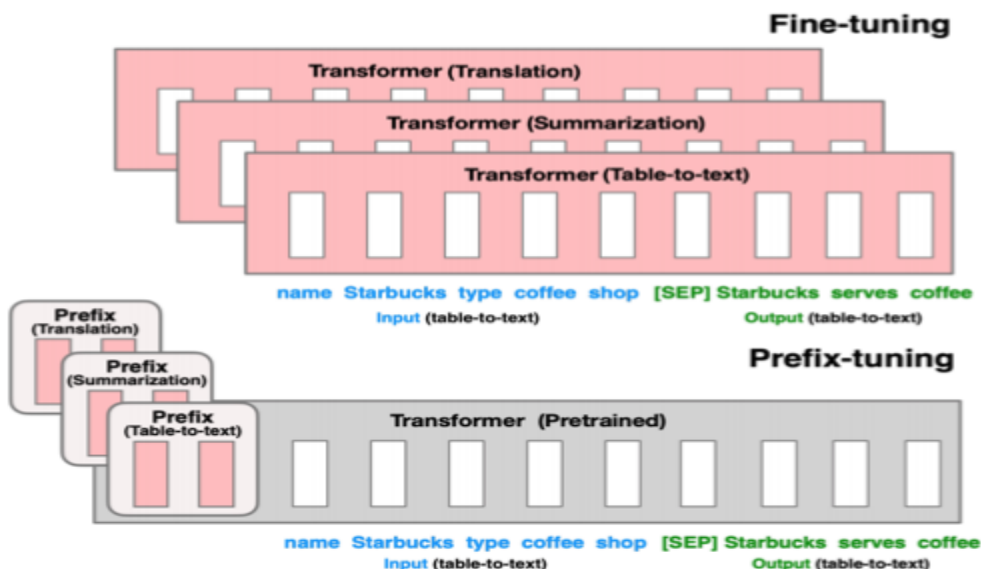
Trong phần này, chúng ta sẽ khám phá một khía cạnh thú vị trong việc tối ưu hóa các embedding cho các tác vụ cụ thể mà không cần phải gắn chặt chúng vào các embedding đã tồn tại. Thay vì phải điều chỉnh lại các embedding về một trạng thái cố định, chúng ta có thể tối ưu hóa các prompt, tức là các embedding của prompt cho một tác vụ cụ thể, và giữ nguyên chúng mà không cần phải lo lắng về việc chúng thực sự là một token trong mô hình.

Điều này có nghĩa là nếu bạn có quyền kiểm soát mô hình của mình, bạn có thể thêm chúng như những phần tử mới trong từ vựng và không gặp vấn đề gì. Nghiên cứu trong bài báo này chỉ ra rằng thay vì phải huấn luyện toàn bộ mô hình với 11 tỷ tham số cho nhiều tác vụ khác nhau trên nhiều tập dữ liệu khác nhau, chúng ta chỉ cần huấn luyện các prompt này, mỗi prompt có khoảng 20K tham số. Tôi không nhớ chính xác độ dài của chúng, có thể là khoảng 10 hoặc 20 tokens.



Điều quan trọng là bạn không cần phải thực hiện học đa tác vụ (multitask learning), tức là không cần phải huấn luyện trên nhiều tác vụ cùng một lúc. Bạn chỉ cần huấn luyện trên một tác vụ duy nhất. Tiếp tục phát triển ý tưởng này, chúng ta chỉ cần huấn luyện các embedding mà bạn đưa vào mô hình.

Prefix Tuning (Li and Liang 2021)



Cách hoạt động của prefix tuning là thay vì chỉ huấn luyện các embeddings đưa vào mô hình, phương pháp này thực sự huấn luyện một tiền tố (prefix) mà bạn sau đó gắn vào mỗi tầng của mô hình. Trong khi "prompt tuning" chỉ thực hiện điều này cho tầng đầu tiên của mô hình, thì "prefix tuning" thực hiện cho mọi tầng của mô hình. Bạn gắn một tiền tố cho mỗi tầng, do đó, đây là một phiên bản biểu đạt hơn của việc prompting.

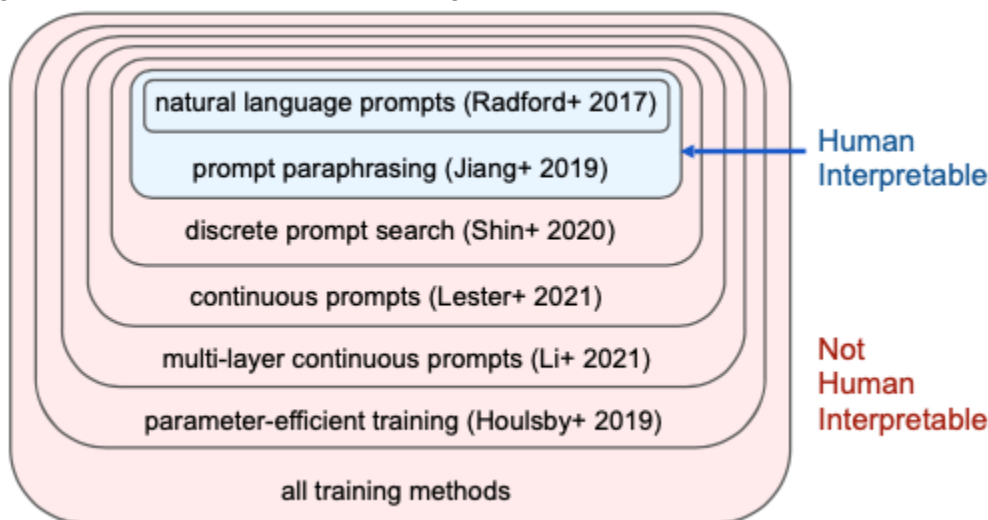
Những phương pháp này là các bước tiến dần từ việc tạo ra một prompt do con người tạo ra đến việc huấn luyện một prompt hoặc một tiền tố cho mô hình.

Prompting and Fine Tuning

Connection to Other Training Methods

Trong bài tiếp theo, tôi sẽ đề cập đến các phương pháp tinh chỉnh hiệu quả tham số, một phiên bản tổng quát hơn của prompt tuning và prefix tuning. Những phương pháp này điều chỉnh một số lượng nhỏ tham số để giúp mô hình thực hiện một nhiệm vụ cụ thể. Ví dụ: LoRA, adapters.

Prompt tuning và prefix tuning thực chất là một phần của lớp tổng quát hơn này. Điều thú vị là chúng ta có thể coi prompting như một hình thức fine-tuning, tức là một cách để tinh chỉnh mô hình nhằm thực hiện một nhiệm vụ cụ thể. Chúng ta có một phân loại các loại prompt trong ngôn ngữ tự nhiên, được tạo ra bởi con người.



Khái niệm "manual prompt engineering" lần đầu tiên được giới thiệu trong bài báo về GPT-2, nơi mà các tác giả chứng minh rằng các mô hình có thể giải quyết nhiệm vụ thông qua cách này. "Prompt paraphrasing" là một bước tiến xa hơn, không còn phụ thuộc vào việc thiết kế prompt thủ công và có thể mở rộng ra một tập hợp rộng hơn các prompt.

Ngoài ra, "discrete prompt search" không nhất thiết phải dựa vào một mô hình paraphrasing; nó có thể sử dụng các mô hình dựa trên gradient hoặc các phương pháp khác để tạo ra các token ngẫu nhiên không phải ngôn ngữ tự nhiên. "Continuous prompts" hay "prompt tuning" là một bước tiến tiếp theo, và "multi-layer continuous prompts" hay "prefix tuning" là một cấp độ cao hơn nữa. Cuối cùng, "parameter efficient tuning" là một khái niệm tổng quát hơn, bao gồm tất cả các phương pháp huấn luyện, bao gồm cả việc fine-tuning mô hình của bạn.

Prompting as a Prior

Khi phương pháp prompting được giới thiệu, nhiều người đã bày tỏ sự hoài nghi, cho rằng các phương pháp này rất "hacky" và không thích việc phải thực hiện kỹ thuật prompt engineering thủ công. Họ cảm thấy nó giống như một "nghệ thuật tối tăm" so với các phương pháp fine-tuning đã được hiểu rõ hơn. Tuy nhiên, tôi lại có cái nhìn tích cực về chúng.

Nếu ai đó quen thuộc với thống kê Bayes hay học máy, chúng ta có khái niệm về "prior probability" (xác suất tiên nghiệm) đối với các tham số và xác suất mà chúng ta nhận được sau khi fine-tuning mô hình. Trong một cách nào đó, các prompt chính là những "prior" tốt đầu tiên cho các mô hình mạng nơ-ron. Chúng cho phép chúng ta xác định nhiệm vụ mà mô hình nên thực hiện hoặc ít nhất là một ý tưởng tổng quát về nhiệm vụ đó trước khi yêu cầu mô hình thực hiện.

Chúng ta có thể sử dụng prior này mà không cần thêm bất kỳ fine-tuning nào, hoặc có thể kết hợp prior mà chúng ta đã cung cấp cho mô hình thông qua mô tả bằng ngôn ngữ tự nhiên về nhiệm vụ với quá trình fine-tuning. Cụ thể, chúng ta có thể khởi tạo phân phối của một prompt bằng cách sử dụng một prompt do con người tạo ra và sau đó tiếp tục fine-tune trên một lượng dữ liệu huấn luyện lớn.

Có một phương pháp được đề xuất bởi Schick and Schütze gọi là "pattern exploiting training", thực hiện chính xác điều này: khởi tạo với một prompt được tạo thủ công và sau đó fine-tune mô hình. Đó là lý do tại sao tôi thích các phương pháp dựa trên prompting. Chúng cung cấp cho chúng ta một cách rất nhanh chóng để tạo ra một hệ thống, đồng thời cho phép chúng ta thực hiện bất kỳ mức độ đào tạo bổ sung nào sau đó.

Resources

1. <https://phontron.com/class/anlp2024/lectures/#prompting-feb-6>
2. [Few-shot Prompting \(Brown et al. 2020\)](#)
3. [Prompt Ordering \(Lu et al. 2021\)](#)
4. [Label Balance and Label Coverage \(Zhang et al. 2022\)](#)
5. [What Makes In-context Learning Work \(Min et al. 2022\)](#)
6. [Chain of Thought \(Wei et al. 2022\)](#)
7. [Let's think Step by Step \(Kojima et al. 2022\)](#)
8. [Structuring Outputs as Programs \(Madaan et al. 2022\)](#)
9. [Program Aided Language Models \(Gao et al. 2022\)](#)
10. [General Tips for Designing Prompts](#)
11. [Prompt Paraphrasing \(Jiang et al. 2019\)](#)
12. [Iterative Prompt Paraphrasing \(Zhou et al. 2021\)](#)
13. [AutoPrompt \(Shin et al. 2020\)](#)
14. [Combining Prompting with Fine-tuning \(Schick and Schütze 2020\)](#)
15. [How to Format Inputs to ChatGPT Models \(OpenAI Cookbook 2024\)](#)