AI VIET NAM – COURSE 2024

Hiểu về LLMs: Tổng quan toàn diện từ việc Huấn luyện đến việc Suy luận

Ngày 30 tháng 3 năm 2024

Ngày công bố:	6/1/2024
Tác giả:	Yiheng Liu, Hao He, Tianle Han, Xu Zhang, Mengyuan Liu, Jiaming Tian, Yutong Zhang, Jiaqi Wang, Xiaohui Gao, Tianyang Zhong, Yi Pan, Shaochen Xu, Zihao Wu, Zhengliang Liu, Xin Zhang, Shu Zhang, Xintao Hu, Tuo Zhang, Ning Qiang, Tianming Liu and Bao Ge
Nguồn:	arXiv
Nguồn dữ liệu (nếu có):	
Từ khóa:	Large Language Models, Training, Inference, Survey
Người tóm tắt:	Hoàng Văn Nhân

I. Tổng quan bài báo:

Dựa trên xu hướng mới nổi là huấn luyện và triển khai LLM với chi phí thấp, bài báo đã xem xét sự phát triển của các kỹ thuật đào tạo mô hình ngôn ngữ lớn và công nghệ triển khai suy luận phù hợp xu hướng đó. Các đối tượng trong từng phần thảo luận của bài báo:

- Phần 1: Lí do viết bài báo.
- Phần 2: Kiến thức nền tảng về LLM.
- Phần 3: Các khía cạnh kĩ thuật của việc huấn luyện LLM.
- Phần 4: Các công nghệ liên quan đến việc suy luận và triển khai LLM.
- Phần 5: Sử dụng LLM.
- Phần 6: Các hướng phát triển trong tương lai và ý nghĩa của chúng.

Bài tóm tắt dưới đây sẽ đi nhanh phần tổng quan 1 và 2 để đến với nội dung chuyên sâu thuộc phần 3 của bài báo.

II. Tóm tắt chi tiết bài báo

1. Tầm quan trọng của việc nghiên cứu cách thức huấn luyện, suy luận hiệu quả cho LLM. Một số kiến thức, khái niệm cần có liên quan đến LLM.

1.1. Tầm quan trọng:

Pre-trained language models (PLM) hay mô hình đào tạo sẵn là một trong những phương pháp chính, phổ biến hiện nay trong lĩnh vực xử lí ngôn ngữ tự nhiên. Những mô hình này thường được huấn luyện trên một tập dữ liệu tổng quát lớn và sau đó sẽ được tinh chỉnh cho một nhiệm vụ cụ thể.

PLM với kích thước tham số lớn hơn đáng kể (6-10 tỷ tham số) và dữ liệu huấn luyện rộng lớn được gọi là Mô hình ngôn ngữ lớn (LLM). Việc có hiểu biết để tự phát triển LLM có thể đóng vai trò thay

thế cho ChatGPT khi mà nó đã không còn là mã nguồn mở, và việc phát triển LLM dành riêng cho từng lĩnh vực đã trở nên rất cần thiết. Và một phần quan trọng trong việc nghiên cứu đó là nghiên cứu cách thức huấn luyện, suy luận hiệu quả cho LLM.

1.2. Kiến thức nền tảng về LLM:

a) Transformer (Bộ chuyển đổi):

Transformer là một mô hình học sâu dựa trên cơ chế chú ý để xử lý dữ liệu chuỗi có thể giải quyết hiệu quả các vấn đề xử lý ngôn ngữ tự nhiên phức tạp. Do tính phù hợp với tính toán song song và độ phức tạp của mô hình, Transformer vượt trội hơn các mạng thần kinh hồi quy (RNN) phổ biến trước đây về độ chính xác và hiệu suất.

Transformer bao gồm Bộ mã hóa (Encoder) và Bộ giải mã (Decoder) cùng cơ chế Chú ý tự thân.

****Cơ chế tự chú ý (Self-Attention):***

Self-Attention là cơ chế giúp Transformers "hiểu" được sự liên quan giữa các từ trong một câu. Nó là một biến thể của cơ chế chú ý, làm giảm sự phụ thuộc vào thông tin bên ngoài và vượt trội trong việc nắm bắt các mối tương quan bên trong trong dữ liệu và giữa các đặc trưng.

Ở trong kiến trúc tổng thể module Multi-head Attention (bản chất là Self-Attention) có 3 mũi tên, đó là 3 vectors Querys (Q) – từ đang được quan tâm, Keys (K) – tất cả các từ trong câu, và Values (V) – lưu trữ thông tin liên quan đến mỗi từ. Từ 3 vectors này ta sẽ tính vector attention cho một từ theo công thức:

$$Attention(Q,\ K,\ V) = softmax\left(\frac{QK^T}{\sqrt{Dimension\ of\ vector\ K}}\right)V$$

Multi-head Attention là cơ chế được mở rộng thêm so với cơ chế Self-Attention bằng cách thực hiện nó nhiều lần một cách song song. Bằng cách này, mô hình có thể nắm bắt cả phần phụ thuộc cục bộ và toàn cục, cho phép hiểu biết toàn diện hơn về chuỗi đầu vào. Sự song song hóa này cũng nâng cao khả năng của mô hình trong việc nắm bắt các mối quan hệ phức tạp giữa các từ.

***Encoder (Bộ mã hóa):**

Module encoder của Transformer bao gồm nhiều lớp con giống hệt nhau.

Cơ chế Multi-head Attention tính toán mức độ quan trọng giữa các vị trí khác nhau trong chuỗi đầu vào để nắm bắt mối liên hệ giữa chúng. Sau đó, mạng thần kinh truyền thẳng (chuyển tiếp) được sử dụng để xử lý và trích xuất thêm các đặc trưng từ đầu ra của cơ chế Multi-head Attention.

Mô-đun encoder dần dần trích xuất các đặc trưng của chuỗi đầu vào thông qua việc xếp chồng nhiều lớp như vậy và chuyển kết quả mã hóa cuối cùng đến mô-đun giải mã để giải mã.

***Decoder (Bộ giải mã):**

Decoder còn có thêm một cơ chế bổ sung là Masked Multi-head Attention (Encoder-Decoder Attention) so với encoder. Quá trình giải mã về cơ bản là giống với quá trình mã hóa, chỉ khác là Decoder giải mã từng từ một và input của Decoder bị masked – nó chỉ được giải mã dựa trên những từ trước đó, không được phép truy cập đến phần thông tin mà nó cần giải mã lúc sau.

***Positional Embedding:**

Khi mỗi thành phần (từ, token) trong một câu đi qua ngăn xếp Bộ mã hóa/Bộ giải mã của Transformer, bản thân mô hình sẽ không có bất kỳ ý nghĩa nào về vị trí/thứ tự cho mỗi token. Kĩ thuật cho phép mô hình có thêm thông tin về vị trí của từng token trong câu nhằm kết hợp tuần tự của các token vào biểu diễn đầu vào được gọi là Positional Embedding.

Trong mô hình Transformer, công thức cốt lõi của Positional Embedding có thể được biểu diễn dưới dạng:

$$PE_{(pos,\ \ 2i)} = \sin\left(\frac{pos}{10000^{\left(\frac{2i}{d_{model}}\right)}}\right) \qquad PE_{(pos,\ \ 2i+1)} = \cos\left(\frac{pos}{10000^{\left(\frac{2i}{d_{model}}\right)}}\right)$$

Trong đó, PE đại diện cho ma trận positional embedding, pos là vị trí của token trong câu, i là chỉ phần tử thứ i trong embedding và d_{model} biểu diễn cho số chiều của lớp ẩn trong mô hình Transformer.

Hai phương thức mã hóa vị trí phổ biến thường được sử dụng trong Transformer là: Mã hóa vị trí tuyệt đối và Mã hóa vị trí tương đối, mỗi phương pháp cung cấp thông tin vị trí theo cách riêng biệt để mô hình có thể xử lý dữ liệu tuần tự một cách hiệu quả.

b) Prompt learning (Học dựa trên lời nhắc):

Prompt learning là một phương pháp học máy, được xây dựng dựa trên các mô hình ngôn ngữ được đào tạo trước và hướng dẫn mô hình thực hiện các nhiệm vụ khác nhau thông qua thiết kế các câu lệnh nhắc, mang lại sự linh hoạt cao hơn cho việc tùy chỉnh các ứng dụng của mô hình.

***Các thành phần và quy trình cơ bản của Prompt learning:**

Prompt learning bao gồm các thành phần cơ bản sau:

- Mẫu Prompt: Là phần chính của prompt, có hai loại mẫu thông dụng:
  - Mẫu "điền vào chỗ trống": loại mẫu chọn một hoặc nhiều vị trí trong văn bản và thể hiện chúng bằng các thẻ [MASK], dùng để nhắc mô hình điền các từ tương ứng.
  - Mẫu "tạo ra dựa trên tiền tố": Việc tạo mẫu dựa trên tiền tố bao gồm việc thêm một tiền tố cụ thể trước một câu để hướng dẫn mô hình tạo văn bản phù hợp.

  Việc lựa chọn mẫu đóng vai trò rất quan trọng trong Prompt learning. Các cách chọn mẫu:
  - Tạo mẫu nhân tạo: là phương pháp trực quan nhất và có hiệu suất tốt trong các ứng dụng thực tế. Tuy nhiên, các mẫu được xây dựng nhân tạo cũng có một số nhược điểm: cần có kiến thức trước khi thiết kế và có thể mắc lỗi.
  - Có hai loại mẫu tạo theo kiểu tự động: mẫu lời nhắc rời rạc và lời nhắc liên tục.

  Có thể sử dụng nhiều mẫu để cải thiện hiệu suất mô hình.
- Ánh xạ câu trả lời: là quá trình chuyển đổi các nhãn (label) của tác vụ thành các từ vựng phù hợp với ngữ cảnh, sau đó chọn đáp án có khả năng cao nhất làm kết quả dự đoán.
- Mô hình ngôn ngữ đã qua huấn luyện (PLMs): Chọn một mô hình ngôn ngữ đã được huấn luyện trước phù hợp để sử dụng làm bộ mã hóa cơ sở.

Quy trình làm việc của Prompt learning chủ yếu gồm 4 phần:

- Sử dụng PLMs làm bộ mã hóa cơ sở.
- Thêm ngữ cảnh (mẫu lời nhắc) với một vị trí [MASK].
- Ánh xạ câu trả lời: chuyển đổi các nhãn (labels) thành các từ nhãn (label words) – verbalizer.
- Phần cầu nối giữa quá trình tiền huấn luyện và tinh chỉnh mô hình (fine-tuning).

***Chiến lược học của Prompt learning:**

Dựa trên nhiệm vụ cụ thể và nhu cầu để lựa chọn chiến lược học phù hợp:

- Tiền huấn luyện rồi tinh chỉnh: là chiến lược phổ biến nhất, phù hợp với hầu hết các nhiệm vụ.
- Không cần tinh chỉnh: phù hợp với các tác vụ đơn giản, chiến lược này có thể giảm đáng kể thời gian đào tạo và mức tiêu thụ tài nguyên tính toán.
- Tinh chỉnh prompt (lời nhắc) với LM (mô hình ngôn ngữ) cố định.

- Tinh chinh LM với prompt cố định.

  Hai chiến lược học phía trên phù hợp với các tác vụ yêu cầu điều khiển chính xác hơn và có thể tối ưu hóa hiệu suất mô hình bằng cách điều chỉnh các tham số prompt hoặc tham số LM.

- Tinh chỉnh prompt kết hợp LM: kết hợp các ưu điểm của cả hai chiến lược trước, có thể cải thiện nhiều hơn nữa hiệu suất mô hình.

Tóm lại, việc sử dụng prompt thiết kế phù hợp và chiến lược học tập hiệu quả sẽ giúp cải thiện hiệu suất mô hình trên các nhiệm vụ khác nhau.

2. Các khía cạnh kĩ thuật của việc huấn luyện LLM:

Việc đào tạo LLM có thể được chia thành ba bước:

- Thu thập rồi xử lý dữ liệu.
- Quá trình tiền huấn luyện: bao gồm việc xác định kiến trúc của mô hình và các nhiệm vụ tiền huấn luyện, đồng thời sử dụng các thuật toán đào tạo song song phù hợp để hoàn thành quá trình huấn luyện.
- Bước thứ ba liên quan đến fine-tuning và căn chỉnh cho phù hợp.

2.1. Chuẩn bị dữ liệu và tiền xử lí dữ liệu:

a) Chuẩn bị tập dữ liệu:

Huấn luyện LLM yêu cầu lượng lớn dữ liệu văn bản và chất lượng của dữ liệu này ảnh hưởng đáng kể đến hiệu suất LLM. Tiền huấn luyện về ngữ liệu quy mô lớn cung cấp cho LLM sự hiểu biết cơ bản về ngôn ngữ và một số khả năng sáng tạo. Các nguồn dữ liệu cho việc tiền huấn luyện thường bao gồm văn bản web, dữ liệu hội thoại, sách và tài liệu chuyên ngành để tăng cường khả năng của LLMs trong các lĩnh vực đó. Có 5 nhóm bộ dữ liệu thường được sử dụng để huấn luyện LLM là:

- Books: Các tập dữ liệu sách như BookCorpus và Gutenberg bao gồm nhiều thể loại văn học khác nhau, từ tiểu thuyết đến lịch sử, khoa học, triết học và nhiều hơn nữa. Chúng được sử dụng rộng rãi trong việc đào tạo các mô hình ngôn ngữ lớn (LLMs) để cung cấp hiểu biết sâu rộng về ngôn ngữ qua các lĩnh vực khác nhau.
- CommonCrawl: Là một kho lưu trữ dữ liệu web crawl lớn, bao gồm hơn 250 tỷ trang web được thu thập trong 16 năm. Dữ liệu từ CommonCrawl chiếm 82% nguồn dữ liệu đào tạo cho GPT-3. Tuy nhiên, do chứa nhiều dữ liệu chất lượng thấp, việc tiền xử lý dữ liệu là cần thiết khi làm việc với CommonCrawl.
- Reddit Links: Reddit là một nền tảng truyền thông xã hội nơi người dùng có thể gửi liên kết và bài đăng, và những người khác có thể bình chọn chúng. Điều này làm cho nó trở thành một nguồn tài nguyên quý giá để tạo ra các tập dữ liệu chất lượng cao.
- Wikipedia: Một dự án bách khoa toàn thư trực tuyến mở rộng với nội dung chất lượng cao về nhiều chủ đề khác nhau. Wikipedia được sử dụng rộng rãi trong việc đào tạo nhiều LLM, đóng vai trò là nguồn tài nguyên quý giá cho việc hiểu và tạo ngôn ngữ.
- Code: Hiện tại có một số lượng hạn chế các tập dữ liệu mã nguồn mở có sẵn công khai. Các nỗ lực hiện tại chủ yếu liên quan đến việc trích xuất mã nguồn có bản quyền mở từ internet, chủ yếu từ Github và Stack Overflow.

b) Tiền xử lí dữ liệu:

Chất lượng xử lý trước dữ liệu ảnh hưởng trực tiếp đến hiệu suất và tính bảo mật của mô hình. Các bước tiền xử lý quan trọng bao gồm:

- Lọc chất lượng: Việc lọc dữ liệu chất lượng thấp thường được thực hiện bằng phương pháp dựa trên quy tắc hoặc phân loại. Các quy tắc có thể bao gồm chỉ giữ lại văn bản chứa số, loại bỏ câu chỉ gồm chữ hoa, và bỏ các tệp có tỷ lệ ký tự và từ vượt quá 0.1.

- Loại bỏ trùng lặp: Mô hình ngôn ngữ đôi khi có thể tạo ra nội dung lặp đi lặp lại do sự lặp lại cao trong dữ liệu huấn luyện. Việc loại bỏ dữ liệu trùng lặp giúp duy trì sự ổn định trong quá trình huấn luyện và cải thiện hiệu suất của LLMs.

- Làm sạch dữ liệu để bảo vệ quyền riêng tư (Privacy scrubbing): Để đảm bảo an ninh mô hình, quá trình tiền xử lý dữ liệu ngôn ngữ cần phải loại bỏ thông tin nhạy cảm, bao gồm các kĩ thuật như ẩn danh, che giấu hoặc mã hóa thông tin cá nhân, địa điểm địa lý và dữ liệu bảo mật khác.

- Lọc văn bản độc hại và thiên kiến: Loại bỏ nội dung độc hại và thiên kiến là một bước quan trọng để phát triển mô hình ngôn ngữ công bằng và không thiên kiến. Điều này đòi hỏi việc áp dụng các kỹ thuật kiểm duyệt nội dung mạnh mẽ để xác định và loại bỏ văn bản có thể duy trì định kiến có hại, ngôn ngữ xúc phạm hoặc quan điểm thiên vị.

2.2. Kiến trúc của mô hình:

Hiện tại, tất cả các LLM đều được xây dựng dựa trên kiến trúc Transformer. Thông thường, kiến trúc PLM thuộc ba loại: Chỉ bộ mã hóa, Bộ giải mã – mã hóa và Chỉ bộ giải mã.

a) Kiến trúc bộ mã hóa – giải mã:

Kiến trúc bộ mã hóa-giải mã bao gồm hai thành phần chính: Bộ mã hóa và bộ giải mã. Encoder chứa nhiều lớp Multi-Head Self-Attention, mã hóa chuỗi đầu vào. Decoder tạo ra chuỗi đích bằng cách sử dụng cross-attention qua biểu diễn đầu ra của Encoder và tạo ra chuỗi đích một cách tự động theo thứ tự.

b) Kiến trúc chỉ bộ giải mã:

LLMs với kiến trúc chỉ bộ giải mã chỉ sử dụng phần decoder của kiến trúc Transformer truyền thống. Mô hình này tạo ra token một cách tuần tự, chú ý đến các token trước đó trong chuỗi. Kiến trúc này thể hiện được tính hiệu quả trong các tác vụ khác nhau như tạo văn bản mà không cần giai đoạn mã hóa rõ ràng. Kiến trúc này có thể được phân loại thành hai loại: kiến trúc Bộ giải mã nguyên nhân (Causal Decoder Architecture) và kiến trúc Bộ giải mã tiền tố (Prefix Decoder Architecture).

- Causal Decoder Architecture: Mỗi token chỉ chú ý đến các token trước đó trong chuỗi đầu vào, sử dụng một cấu hình mặt nạ đặc biệt để thực hiện quá trình giải mã một chiều. Kiến trúc này được áp dụng trong các mô hình như GPT series.

- Prefix Decoder Architecture: Kết hợp ưu điểm của cả hai kiến trúc Encoder-decoder và Causal Decoder, cho phép sự chú ý hai chiều đối với các token tiền tố trong khi vẫn duy trì sự chú ý một chiều khi tạo ra các token tiếp theo. Kiến trúc này được sử dụng trong các mô hình như PaLM và GLM.

Cả hai kiến trúc này đều tập trung vào việc tạo ra chuỗi văn bản một cách tự động, từng token một, dựa trên ngữ cảnh được cung cấp bởi các token trước đó.

2.3. Nhiệm vụ tiền huấn luyện:

Mô hình ngôn ngữ lớn (LLM) thường học cách biểu diễn ngôn ngữ phong phú thông qua quá trình tiền huấn luyện. Trong quá trình đào tạo trước, các mô hình này tận dụng kho dữ liệu mở rộng và trải qua quá trình đào tạo thông qua các phương pháp học tập tự giám sát. Một trong số đó là Mô hình hóa ngôn ngữ (LM), LM tiền huấn luyện thông qua việc dự đoán từ tiếp theo trong một ngữ cảnh cho trước, giúp mô hình học được cách sử dụng từ vựng, ngữ pháp, và ngữ nghĩa. Quá trình học dần dần này cho phép mô hình nắm bắt các mẫu và thông tin vốn có trong ngôn ngữ, mã hóa một lượng lớn kiến thức ngôn ngữ thành các tham số của nó. Sau khi quá trình đào tạo trước hoàn tất, các tham số

mô hình này có thể được tinh chỉnh cho các tác vụ xử lý ngôn ngữ tự nhiên khác nhau để thích ứng với các yêu cầu tác vụ cụ thể.

Ngoài mô hình hóa ngôn ngữ, có các nhiệm vụ tiền huấn luyện khác như sử dụng văn bản với một số phần được thay thế ngẫu nhiên, sau đó sử dụng phương pháp tự động hồi quy để phục hồi các phần đã thay thế.

2.4. Huấn luyện mô hình:

a) Huấn luyện song song:

Huấn luyện song song là một phương pháp huấn luyện mô hình học máy, đặc biệt là các LLM bằng cách sử dụng nhiều bộ xử lý hoặc GPU cùng một lúc để xử lý dữ liệu và tham số mô hình. Mục tiêu của parallel training là giảm thời gian cần thiết để huấn luyện mô hình và tăng khả năng mở rộng của mô hình khi xử lý dữ liệu lớn và phức tạp.

Giao tiếp tập thể (collective communication) là một khái niệm quan trọng trong huấn luyện song song. giao tiếp tập thể liên quan đến việc trao đổi dữ liệu giữa các bộ xử lý hoặc GPU trong quá trình huấn luyện mô hình. Các hoạt động giao tiếp tập thể bao gồm:

- Broadcast: Gửi dữ liệu từ một GPU đến tất cả các GPU khác.
- Reduce: Tổng hợp dữ liệu từ tất cả các GPU và gửi kết quả đến một GPU.
- All Reduce: Tổng hợp dữ liệu từ tất cả các GPU và phân phối kết quả đến tất cả các GPU.
- Reduce Scatter: Tổng hợp dữ liệu từ tất cả các GPU và phân phối các phần dữ liệu đến từng GPU.
- All Gather: Thu thập dữ liệu từ tất cả các GPU và phân phối đến tất cả các GPU.

Trong huấn luyện song song có 4 phương pháp chính:

- Data Parallel: Sử dụng một máy chủ tham số để lưu trữ các tham số của mô hình và toàn bộ batch dữ liệu. Các GPU đồng bộ hóa tham số mô hình và chia dữ liệu thành từng phần, mỗi GPU xử lý một phần dữ liệu. Sau đó, các gradient được tổng hợp và gửi trở lại máy chủ tham số.
- Model Parallel: Phân chia mô hình thành các phần nhỏ hơn và mỗi GPU sẽ xử lý một phần của mô hình. Điều này giúp xử lý các mô hình lớn không vừa với bộ nhớ của một GPU đơn lẻ.
- ZeRO: Tối ưu hóa việc sử dụng bộ nhớ và băng thông mạng bằng cách loại bỏ sự trùng lặp của dữ liệu trên các GPU, giúp mở rộng quy mô huấn luyện mà không tăng cường độ sử dụng bộ nhớ.
- Pipeline Parallel: Chia nhỏ quá trình huấn luyện thành các giai đoạn và mỗi GPU sẽ xử lý một giai đoạn cụ thể. Điều này giúp tăng hiệu suất bằng cách giảm thời gian chờ đợi giữa các GPU.

b) Huấn luyện chính xác hỗn hợp:

Huấn luyện độ chính xác hỗn hợp (Mixed Precision Training) là một kỹ thuật trong huấn nhằm tăng tốc độ tính toán và giảm bớt việc sử dụng bộ nhớ. Kỹ thuật này sử dụng cả số thực dấu phẩy động 16-bit (FP16) và 32-bit (FP32) trong quá trình huấn luyện.

FP16 có phạm vi số học nhỏ hơn và độ chính xác thấp hơn so với FP32, nhưng lại cho phép tính toán nhanh hơn. Để cải thiện tốc độ tính toán, chúng ta có thể chuyển từ FP32 sang FP16, vì số lượng tham số trong một mô hình thường nằm trong phạm vi số của FP16. Khi cập nhật tham số, việc nhân gradient với tốc độ học có thể dẫn đến mất mát do tràn dấu phẩy động nếu sử dụng FP16, vì vậy tham số cập nhật được biểu diễn dưới dạng FP32. Trong trình tối ưu hóa, lượng cập nhật được lưu dưới dạng FP32 và tích lũy nó một cách hiệu quả thông qua một tham số FP32 tạm thời, sau đó được chuyển đổi trở lại thành các tham số FP16.

c) Offloading:

Offloading là quá trình chuyển các tham số của trình tối ưu hóa từ GPU sang CPU để giảm tải tính toán, sử dụng ZeRO3 để giảm kích thước của tham số, gradient và trình tối ưu hóa xuống 1/n số lượng GPU.

d) Overlapping:

Overlapping là một kỹ thuật tối ưu hóa hiệu suất bằng cách thực hiện các phép tính một cách không đồng bộ. Điều này cho phép mô hình tiếp tục thực hiện các phép tính khác trong khi đang chờ xử lý một yêu cầu bộ nhớ, giúp tận dụng tối đa khả năng tính toán của hệ thống và tối ưu hóa quá trình lan truyền chuyển tiếp trong huấn luyện mô hình.

e) Checkpoint:

Sử dụng cơ chế checkpoint không lưu trữ tất cả kết quả trung gian trong bộ nhớ GPU mà chỉ giữ lại các điểm kiểm tra nhất định, giảm lượng bộ nhớ cần thiết và cho phép tái tính toán trong quá trình lan truyền ngược.

2.5. Fine-tuning (tinh chỉnh):

Có 3 loại kĩ thuật tinh chỉnh: tinh chỉnh có giám sát (SFT), điều chỉnh phù hợp (alignment tuning), tinh chỉnh tham số hiệu quả.

a) Tinh chỉnh có giám sát:

SFT là điều chỉnh tiếp mô hình đã được đào tạo trước, dựa trên tập dữ liệu được gắn nhãn cho một nhiệm vụ cụ thể (ví dụ: dịch thuật, trả lời câu hỏi).

Điều chỉnh hướng dẫn (Instruction tuning): Một kỹ thuật SFT phổ biến trong đó LLM được đào tạo thêm trên một bộ dữ liệu gồm các cặp "(hướng dẫn, đầu ra)" để cải thiện khả năng kiểm soát bằng cách hiểu và tuân theo hướng dẫn của con người.

b) Điều chỉnh phù hợp:

Điều chỉnh phù hợp là việc điều chỉnh để đảm bảo rằng mô hình tạo ra kết quả:

- Hữu ích: Mô hình cần cung cấp thông tin có giá trị, phục vụ tốt cho nhu cầu hoặc mục tiêu của người dùng.
- Trung thực: Mô hình phải tạo ra thông tin chính xác, không làm sai lệch sự thật, giữ được niềm tin của người dùng.
- Không gây hại: Mô hình không được tạo ra nội dung có hại, xúc phạm hoặc nguy hiểm, đảm bảo an toàn cho người dùng và xã hội.

Phương pháp: Sử dụng phương pháp Học Tăng Cường với Phản Hồi Từ Con Người (RLHF), thu thập dữ liệu phản hồi từ con người để huấn luyện một mô hình phần thưởng (RM), sử dụng trong quá trình học tăng cường để tinh chỉnh LLM.

c) Tinh chỉnh tham số hiệu quả:

Kỹ thuật tinh chỉnh hiệu quả tham số giúp giảm đáng kể chi phí tính toán và bộ nhớ, bằng cách chỉ tinh chỉnh một phần nhỏ hoặc bổ sung của các tham số mô hình. Các phương pháp phổ biến như Low-Rank Adaptation (LoRA), Prefix Tuning và P-Tuning cho phép tinh chỉnh mô hình một cách hiệu quả ngay cả trong môi trường có hạn chế về tài nguyên.

d) Tinh chỉnh an toàn:

Tăng cường an toàn và trách nhiệm của các mô hình ngôn ngữ lớn (LLMs) bằng cách tích hợp thêm kỹ thuật an toàn trong quá trình tinh chỉnh. Điều này bao gồm cả 3 kĩ thuật chính.

- Tinh chỉnh an toàn có giám sát: Dùng dữ liệu do chuyên gia tạo ra để huấn luyện LLM nhận diện và tránh nội dung nguy hiểm.
- RLHF an toàn: Khen thưởng LLM khi đưa ra phản hồi an toàn nhất, ví dụ như từ chối tạo nội dung độc hại.

- Lọc ngữ cảnh an toàn: Huấn luyện LLM dựa trên dữ liệu an toàn được tạo ra bằng cách thêm lời nhắc an toàn vào đầu lời nhắc nguy hiểm.

2.6. Đánh giá:

a) Bộ dữ liệu kiểm tra:

Quá trình đánh giá phụ thuộc rất nhiều vào các bộ dữ liệu thích hợp để đánh giá khả năng LLM. Một số bộ dữ liệu thường được sử dụng là:

- Dành cho các mô hình hình ảnh**:** ImageNet (thị giác máy tính), Open Images.
- Dành cho các mô hình dựa trên văn bản**:** GLUE, SuperGLUE, MMLU (khả năng chung), CMMLU (tiếng Trung), XTREME/XTREME-R (đa ngôn ngữ)
- Dành cho các nhiệm vụ cụ thể**:** MATH/GSM8K (toán học), HumanEval/MBPP (tạo mã), HelloSwag/PIQA/ BoolQ/v.v. (lý giải thông thường), MedQA-USMLE/MedMCQA (tri thức về y khoa).

b) Đánh giá khả năng trả lời câu hỏi mở (ODQA):

Đánh giá ODQA là quan trọng vì nó ảnh hưởng đến trải nghiệm người dùng. Các bộ dữ liệu phổ biến bao gồm SquAD và Natural Questions, với F1 và EM là chỉ số đánh giá. Tuy nhiên, đánh giá của con người vẫn cần thiết do những hạn chế trong phương pháp ghép từ.

c) Đánh giá an ninh:

LLM yêu cầu đánh giá an ninh cẩn thận để giải quyết các mối đe dọa và lỗ hổng tiềm ẩn:

- Thiên kiến tiềm ẩn: Đánh giá an ninh cần xem xét liệu mô hình có tạo ra hoặc tăng cường những định kiến như giới tính hay chủng tộc không và cách giảm bớt hoặc sửa chữa chúng.
- Bảo vệ quyền riêng tư: Cần đảm bảo bảo vệ quyền riêng tư dữ liệu người dùng hiệu quả để ngăn chặn rò rỉ và bị lạm dụng.
- Tấn công thù địch: LLM có thể dễ bị thao túng thông qua các cuộc tấn công thù địch. Giải quyết các lỗ hổng này là rất quan trọng.

d) Phương pháp đánh giá:

Nên kết hợp đánh giá tự động và thủ công để đánh giá toàn diện hiệu suất mô hình ngôn ngữ:

- Đánh giá tự động: Các chỉ số như BLEU, ROUGE và BERTScore cung cấp các phương pháp định lượng đo lường hiệu suất. Chúng hiệu quả khi phân tích dữ liệu quy mô lớn nhưng không thể hiểu hết được sự phức tạp của ngôn ngữ.
- Đánh giá thủ công: Các chuyên gia đánh giá chủ quan chất lượng đầu ra LLM. Phương pháp này có giá trị cho các nhiệm vụ cụ thể và xác định các vấn đề, lỗi tinh vi mà đánh giá tự động có thể bỏ qua, nhưng nó tốn thời gian và chủ quan.

2.7. LLM Framework:

LLM Framework cung cấp các công nghệ và phương pháp tiên tiến để xử lý số lượng lớn tham số của LLMs, giúp tối ưu hóa quá trình huấn luyện và cải thiện hiệu suất của mô hình trên các tài nguyên tính toán hạn chế.

Một số LLM Framework sử dụng công nghệ đào tạo phân tán tận dụng GPU, CPU và bộ nhớ NVMe là:

- Transformers: Thư viện mã nguồn mở của Hugging Face, cung cấp API thân thiện để tùy chỉnh các mô hình đã được huấn luyện trước sử dụng kiến trúc Transformer.

- DeepSpeed: Phát triển bởi Microsoft, thư viện này hỗ trợ công nghệ ZeRO cho việc huấn luyện LLM hiệu quả với các tính năng như phân chia trạng thái tối ưu hóa và huấn luyện chính xác hỗn hợp tùy chỉnh.
- BMTrain: Bộ công cụ của Đại học Thanh Hoa cho phép huấn luyện phân tán các mô hình lớn một cách đơn giản mà không cần tái cấu trúc mã, hỗ trợ các kỹ thuật tối ưu hóa khác nhau.
- Megatron-LM: Thư viện của NVIDIA cho việc huấn luyện các mô hình ngôn ngữ quy mô lớn, sử dụng song song hóa mô hình/dữ liệu và huấn luyện chính xác hỗn hợp để sử dụng GPU một cách hiệu quả.

3. Suy luận (Inference) với các mô hình ngôn ngữ lớn:

LLM đang ngày càng trở nên mạnh mẽ, nhưng quy mô lớn của chúng dẫn đến nhu cầu đáng kể về tài nguyên. Dưới đây là một số kỹ thuật để tối ưu hóa việc suy luận của LLM nhằm giảm chi phí tính toán và sử dụng bộ nhớ.

3.1. Nén mô hình:

- Chắt lọc kiến thức: Chuyển tải kiến thức từ mô hình cồng kềnh sang mô hình nhỏ hơn, phù hợp hơn cho việc triển khai.
- Cắt tỉa mô hình: Loại bỏ các phần dư thừa khỏi ma trận tham số của các mô hình lớn. Có 2 cách thực hiện:
  - Cắt tỉa không có cấu trúc: loại bỏ các kết nối hoặc trọng số riêng lẻ trong mạng lưới thần kinh mà không tuân theo bất kỳ mẫu cấu trúc cụ thể nào.
  - Cắt tỉa có cấu trúc: chỉ loại bỏ các mẫu cấu trúc cụ thể.
- Lượng tử hóa mô hình (gọi như vậy vì ta phải biến hóa các đối tượng vốn có thành đối tượng cơ bản nhất như trong vật lí là lượng tử): Giảm số lượng bit được sử dụng để tính toán số, giảm yêu cầu lưu trữ và tính toán. Tuy nhiên, có một sự đánh đổi giữa độ chính xác và hiệu suất. Ví dụ: từ dạng số thực sang số nguyên.
- Chia sẻ trọng số: Sử dụng cùng một bộ tham số cho nhiều phần của LLM. Điều này giúp giảm số lượng tham số cần học, tăng hiệu quả tính toán và giảm nguy cơ quá khớp.
- Phép xấp xỉ hạng thấp: Tạo ra các mô hình nhỏ gọn hơn với ít tham số hơn bằng cách phân tách các ma trận lớn thành các ma trận nhỏ hơn, cho phép suy luận hiệu quả trên các thiết bị bị hạn chế tài nguyên.

3.2. Lập lịch bộ nhớ:

Tối ưu hóa việc sử dụng bộ nhớ trong quá trình suy luận LLM bằng cách quản lý các mẫu truy cập bộ nhớ một cách hiệu quả. Điều này rất quan trọng vì các mô hình lớn thường có kiến trúc phức tạp với yêu cầu bộ nhớ đáng kể. Một ví dụ là BMInf, tận dụng các nguyên tắc bộ nhớ ảo để lên lịch hiệu quả các tham số của mỗi lớp giữa GPU và CPU.

3.3. Song song hóa:

Sử dụng nhiều đơn vị xử lý để phân phối khối lượng công việc suy luận LLM, cải thiện thông lượng tổng thể và giảm độ trễ.

- Song song hóa dữ liệu: Phân phối dữ liệu trên nhiều GPU để xử lý nhanh hơn.
- Song song hóa tensor: Tham số của mô hình được chia thành nhiều tensor và mỗi tensor được tính toán trên các đơn vị xử lý khác nhau.

- Song song hóa pipeline: Chia nhỏ tính toán thành các giai đoạn được xử lý bởi các GPU khác nhau, cho phép hỗ trợ cho các mô hình lớn hơn và nâng cao khả năng sử dụng thiết bị.

3.4. Tối ưu hóa cấu trúc:

Tập trung vào việc giảm thiểu truy cập bộ nhớ trong quá trình suy luận LLM để cải thiện tốc độ. Các kỹ thuật như FlashAttention và PagedAttention nâng cao tốc độ tính toán bằng cách sử dụng phương pháp tính toán theo khối, giảm thiểu chi phí lưu trữ liên quan đến ma trận.

3.5. Inference Framework:

Cung cấp nền tảng để triển khai và chạy LLM. Các kỹ thuật như tính toán song song, nén mô hình, lập lịch bộ nhớ và tối ưu hóa cấu trúc transformer đã được tích hợp hiệu quả vào các inference framework chính. Các framework này cung cấp các công cụ và giao diện giúp hợp lý hóa các quy trình triển khai và suy luận cho các trường hợp sử dụng khác nhau.

4. Việc sử dụng các mô hình ngôn ngữ lớn:

LLMs có thể được áp dụng trong hầu hết các lĩnh vực chuyên môn. Sau khi được huấn luyện sơ bộ và tinh chỉnh, LLMs chủ yếu được sử dụng bằng cách thiết kế các lời nhắc phù hợp cho nhiều nhiệm vụ khác nhau qua một số khả năng sau: Zero-shot, few-shot, kết hợp các lời nhắc thành một chuỗi suy nghĩ giúp cải thiện học tập về ngữ cảnh,...

Chiến lược triển khai LLM:

- Truy cập API: Sử dụng khả năng của các mô hình độc quyền, mạnh mẽ được cung cấp thông qua các API mở (ví dụ: ChatGPT).
- Triển khai LLM mã nguồn mở để sử dụng cục bộ.
- Tinh chỉnh LLM mã nguồn mở để đáp ứng các tiêu chuẩn cụ thể của lĩnh vực nhất định và sau đó triển khai chúng cục bộ cho các ứng dụng chuyên biệt.

5. Các hướng phát triển trong tương lai và ý nghĩa của chúng:

Xu hướng phát triển LLM trong tương lai:

- Mở rộng mô hình: Dự kiến LLMs sẽ tiếp tục mở rộng quy mô, nâng cao hiệu suất và khả năng học hỏi.
- Phát triển đa phương tiện: LLMs tương lai có thể xử lý không chỉ văn bản mà còn cả hình ảnh, video và âm thanh, mở rộng ứng dụng của chúng.
- Hiệu quả: Tập trung vào giảm chi phí đào tạo và suy luận thông qua các kỹ thuật như chắt lọc kiến thức, nén mô hình.
- Đào tạo cho ngành cụ thể: Tùy chỉnh LLMs cho các ngành cụ thể để hiểu rõ hơn về thuật ngữ và bối cảnh đặc thù của ngành.
- Kiến trúc mới: Khám phá các kiến trúc thay thế như RNN cho LLM là một hướng nghiên cứu hấp dẫn, có khả năng giải quyết các hạn chế như yêu cầu đầu vào kích thước cố định trong transformer.

Đối với các nhà nghiên cứu AI, sự hợp tác giữa các ngành và lĩnh vực khác nhau đang trở nên thiết yếu. Họ phải triển kỹ năng kỹ thuật để giải quyết những thách thức trong phát triển LLMs. Các vấn đề đạo đức và ảnh hưởng xã hội của LLMs cũng là những lĩnh vực quan trọng cần nghiên cứu và cải thiện liên tục.