

**BỘ GIÁO DỤC VÀ ĐÀO
TẠO**

**BỘ NÔNG NGHIỆP VÀ
PTNT**

TRƯỜNG ĐẠI HỌC THỦY LỢI



TƯỞNG ĐĂNG VƯƠNG QUỐC

**XÂY DỰNG MÔ HÌNH HỌC SÂU TRONG NGÔN NGỮ TỰ NHIÊN CHO
NHIỆM VỤ TRẢ LỜI CÂU HỎI TRẮC NGHIỆM**

ĐỒ ÁN TỐT NGHIỆP

HÀ NỘI, NĂM 2024

BỘ GIÁO DỤC VÀ ĐÀO TẠO BỘ NÔNG NGHIỆP VÀ PTNT
TRƯỜNG ĐẠI HỌC THỦY LỢI

TƯỜNG ĐĂNG VƯƠNG QUỐC

**XÂY DỰNG MÔ HÌNH HỌC SÂU TRONG NGÔN NGỮ TỰ NHIÊN CHO
NHIỆM VỤ TRẢ LỜI CÂU HỎI TRẮC NGHIỆM**

Ngành: Công nghệ thông tin

Mã số: 7480201

NGƯỜI HƯỚNG DẪN: 1. ThS. Trương Xuân Nam

HÀ NỘI, NĂM 2024



CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập - Tự do - Hạnh phúc



NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

Họ tên sinh viên: Trương Đăng Vương Quốc

Hệ đào tạo : Chính quy

Lớp: 61TH-NB

Ngành: Công nghệ thông tin

Khoa: Công nghệ thông tin

1. TÊN ĐỀ TÀI:

Xây dựng mô hình học sâu trong ngôn ngữ tự nhiên cho nhiệm vụ trả lời câu hỏi trắc nghiệm

2. CÁC TÀI LIỆU CƠ BẢN:

3. CÁC TÀI LIỆU THAM KHẢO

4. NỘI DUNG CÁC PHẦN THUYẾT MINH VÀ TÍNH TOÁN:

Nội dung thuyết minh và tính toán	Tỷ lệ %
Chương 1: Giới thiệu bài toán, giới thiệu tổng quan	15%

Chương 2: Các kĩ thuật sử dụng trong bài toán về NLP.	25%
Chương 3: Ứng dụng cho nhiệm vụ trả lời câu hỏi trắc nghiệm.	25%
Chương 4: Kết quả và đánh giá	25%
Kết luận	10%

5 - GIÁO VIÊN HƯỚNG DẪN TỪNG PHẦN:

Giáo viên hướng dẫn toàn bộ quá trình thực hiện đồ án: **ThS. Trương Xuân Nam**

6. NGÀY GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

Ngày tháng năm 2024

Trưởng Bộ môn

(Ký và ghi rõ Họ tên)

Giáo viên hướng dẫn chính

(Ký và ghi rõ Họ tên)

Nhiệm vụ Đồ án tốt nghiệp đã được Hội đồng thi tốt nghiệp của Khoa thông qua

Ngày tháng năm 2024

Chủ tịch Hội đồng

(Ký và ghi rõ Họ tên)

Sinh viên đã hoàn thành và nộp bản Đồ án tốt nghiệp cho Hội đồng thi ngày... tháng...
năm 2024

Sinh viên làm Đồ án tốt nghiệp

(Ký và ghi rõ Họ tên)



TRƯỜNG ĐẠI HỌC THỦY LỢI

KHOA CÔNG NGHỆ THÔNG TIN

BẢN TÓM TẮT ĐỀ CƯƠNG ĐỒ ÁN TỐT NGHIỆP

TÊN ĐỀ TÀI: Xây dựng mô hình học sâu trong ngôn ngữ tự nhiên cho nhiệm vụ trả lời câu hỏi trắc nghiệm.

Sinh viên thực hiện: Tưởng Đăng Vương Quốc

Lớp : 61THNB

Email: tuongdangvuongquoc901@gmail.com

Điện thoại : 0988965602

Giáo viên hướng dẫn : Thầy Trương Xuân Nam

TÓM TẮT ĐỀ TÀI

Nhiệm vụ của bài toán được đề xuất bởi cuộc thi của kaggle có tên “Kaggle-LLM science exam” với mục tiêu ứng dụng mô hình ngôn ngữ lớn (large language model) trả lời câu hỏi khoa học khó. Mục tiêu của cuộc thi lấy cảm hứng từ tập dữ liệu openbookQA, cuộc thi này thách thức người chơi tham gia trả lời các câu hỏi khó dựa trên cơ sở khoa học được viết bởi mô hình ngôn ngữ lớn.

Nhiệm vụ trả lời câu hỏi trắc nghiệm trong mô hình hiện nay đóng vai trò quan trọng trong nhiều ứng dụng của trí tuệ nhân tạo ngày nay. Trong mô hình học sâu trả lời câu hỏi trắc nghiệm đòi hỏi mô hình phải có khả năng hiểu và phân tích nội dung của câu hỏi từ đó tìm ra câu trả lời chính xác.

Nhiệm vụ này bao gồm các bước:

- Tiền xử lý dữ liệu : Vector hóa văn bản làm đầu vào cho mô hình học sâu.
Hiểu câu hỏi : Mô hình phải học được cách hiểu được câu hỏi và câu trả lời.
- Tìm câu trả lời : Mô hình phải dựa vào thông tin đã học đưa ra kết quả có xác suất chính xác cao nhất là đáp án cho mô hình.

Nhiệm vụ trả lời câu hỏi trắc nghiệm trong mô hình học sâu có rất nhiều ứng dụng thực tế, chẳng hạn như hệ thống trả lời câu hỏi tự động, trợ lý ảo và nhiều ứng dụng khác trong lĩnh vực y tế, giáo dục, kỹ thuật... Mô hình học sâu ngày càng phát triển và hoàn thiện, giúp tối ưu hóa hiệu suất và tạo ra các ứng dụng tiện ích mạnh mẽ cho xã hội. Từ đó em quyết định chọn đề tài **“Xây dựng mô hình học sâu trong ngôn ngữ tự nhiên cho nhiệm vụ trả lời câu hỏi trắc nghiệm.”**

Công nghệ sử dụng:

- Ngôn ngữ python.
- Học sâu cho ngôn ngữ tự nhiên.
- Fine-tuning pretrained model

CÁC MỤC TIÊU CHÍNH

Mục tiêu 1. Tìm hiểu bài toán trả lời câu hỏi trắc nghiệm

Mục tiêu 2. Lựa chọn mô hình pretrained phù hợp cho bài toán

Mục tiêu 3. Tăng độ chính xác của mô hình bằng cách tìm kiếm thông tin câu hỏi từ wikipedia

Mục tiêu 4. Đánh giá mô hình

Mục tiêu 5. Kết luận

KẾT QUẢ DỰ KIẾN

- Hoàn thành các mục tiêu đã đề ra
- Xây dựng được mô hình học máy trả lời câu hỏi trắc nghiệm
- Báo cáo tổng kết Đồ Án Tốt Nghiệp

TÀI LIỆU THAM KHẢO

<https://www.kaggle.com/competitions/kaggle-llm-science-exam/data>

https://huggingface.co/docs/transformers/v4.31.0/en/tasks/multiple_choice

LỜI CAM ĐOAN

Em xin cam đoan đây là Đồ án tốt nghiệp của cá nhân em. Các kết quả trong Đồ án tốt nghiệp này là trung thực và thực hiện dưới sự hướng dẫn của thầy ThS.Trương Xuân Nam. Việc tuân thủ quy định về trích dẫn và ghi nguồn tài liệu tham khảo đã được thực hiện sau khi tham khảo các nguồn tài liệu (nếu có).

Tác giả ĐATN

Quốc

Tưởng Đăng Vương Quốc

LỜI CẢM ƠN

Qua quãng thời gian học tập, rèn luyện và nghiên cứu tại Khoa Công Nghệ Thông Tin - Trường Đại Học Thủy Lợi, bản thân em đã trải qua một chặng đường học với một môi trường đào tạo chất lượng, được hướng dẫn bởi những thầy cô tận tâm và yêu quý.

Em xin gửi lời cảm ơn chân thành đến quý thầy cô tại Trường Đại học Thủy Lợi, người đã trang bị cho em những kiến thức vô cùng quý báu suốt bốn năm học tập tại đây. Đặc biệt, lòng nhiệt huyết và sự chuyên nghiệp của các thầy cô trong Khoa Công nghệ thông tin đã tận tình giảng dạy, chỉ bảo và trang bị cho em những kiến thức chuyên ngành vô cùng quý giá, giúp em có đủ nền tảng để hoàn thành Đồ án tốt nghiệp này. Em chân thành biết ơn sự đồng hành và sự hỗ trợ quý báu từ các thầy cô trong suốt hành trình học tập của mình.

Em muốn bày tỏ lòng biết ơn sâu sắc đặc biệt đến TS. Trương Xuân Nam, giảng viên tận tình của Trường Đại học Thủy Lợi. Từ những ngày đầu tiên xây dựng đồ án, thầy đã không ngừng hướng dẫn em, đưa ra những góp ý chi tiết và truyền đạt những kinh nghiệm quý báu. Những đóng góp này đã giúp em vượt qua những thách thức và hoàn thành Đồ án tốt nghiệp một cách xuất sắc. Em xin gửi lời cảm ơn sâu sắc nhất đến TS. Trương Xuân Nam vì sự tận tâm và sự hỗ trợ quý báu trong quá trình nghiên cứu và thực hiện đồ án của mình.

Qua quá trình hoàn thành đồ án, em nhận thức rõ rằng kiến thức của mình vẫn còn hạn chế và thiếu kinh nghiệm thực tế. Điều này là không tránh khỏi và em tự nhận ra những điểm chưa hoàn hảo. Vì lẽ đó, em chân thành kính mong nhận được những ý kiến đóng góp quý báu từ quý thầy cô để em có cơ hội hoàn thiện và nâng cao sâu rộng kiến thức của mình trong lĩnh vực này.

Em xin chân thành cảm ơn vì sự quan tâm và hỗ trợ từ phía quý thầy cô!

MỤC LỤC

LỜI CAM ĐOAN	i
DANH MỤC HÌNH VẼ.....	vi
Chương 1: Giới thiệu bài toán, giới thiệu tổng quan	1
1.1.Giới thiệu tổng quan	1
1.1.1.Ngữ cảnh lịch sử	1
1.2.Lý do lựa chọn đề tài.....	3
1.2.1.Thách thức trong xử lý câu hỏi trắc nghiệm	3
1.2.2.Sự cần thiết cho giáo dục và doanh nghiệp.....	3
1.3.Mục tiêu nghiên cứu	3
1.4.Phạm vi nghiên cứu.....	4
Chương 2:Các kĩ thuật sử dụng trong bài toán về NLP.....	4
2.1. Deep learning.....	4
2.2. Neural network.....	6
2.2.1. Perceptrons.....	6
2.2.2. Multi-Layer Perceptron.....	9
2.3.Tokenization.....	10
2.4.Word Embedding và Vectorization	14
2.5.Pre-trained model.....	15
2.6. Transformer-Encoder	15
2.6.1.Embedding layer with Position Encoding.....	16
2.6.2.Encoder	18
2.7.Activation GELU	22
2.8.Tính toán độ tương đồng cosin, L2	24
2.9.Phương pháp đánh giá.....	25
2.9.1.Phương pháp đánh giá của cuộc thi	25
2.9.2.Accuracy	26
Chương 3:Ứng dụng cho nhiệm vụ trả lời câu hỏi trắc nghiệm	27
3.1. Giới thiệu	27
<u>3.2.Ứng dụng Pre-trained model cho nhiệm vụ trả lời câu hỏi trắc nghiệm</u>	<u>28</u>

3.3. Các công cụ xây dựng.....	29
3.3.1.Sử dụng ngôn ngữ python.....	29
3.3.2.Sử dụng công cụ Kaggle notebooks.....	29
3.3.3.Các thư viện sử dụng.....	30
3.4.Giới thiệu về tập dữ liệu.....	31
3.5.Xây dựng mô hình trả lời câu hỏi trắc nghiệm	36
3.5.1.Các phương pháp và mô hình đã thử nghiệm	36
3.5.2. Chiến lược cải thiện kết quả của mô hình.....	38
3.5.3.Quy trình trả lời câu hỏi trắc nghiệm	39
Chương 4: Kết quả và đánh giá.....	40
4.1.Xử lý dữ liệu	40
4.2.Xây dựng hàm đánh giá cho quá trình huấn luyện.....	40
4.3.Xây dựng model.....	41
4.4.Kết quả đào tạo	43
4.5.Kết quả của quá trình tham gia cuộc thi.....	46
Kết luận.....	48
Tài liệu tham khảo	50

DANH MỤC CHỮ VIẾT TẮT

NLP: Natural Language Processing

LLM: Large Language Model

NLLLoss : Negative Log Likelihood Loss

RAG: Retrieval-Augmented Generation

DANH MỤC HÌNH VẼ

Hình 1 Hình ảnh học sâu trong trí tuệ nhân tạo.....	4
Hình 2 Cấu tạo một Perceptron	7
Hình 3 Đồ thị của hàm với bias	8
Hình 4 kiến trúc mạng neural network	9
Hình 5 Kiến trúc transformer	16
Hình 6 Encoders input	17
Hình 7 Sự thay đổi về tần số.....	18
Hình 8. Cấu tạo lớp encoder.....	19
Hình 9. Attention và multi-head-attention.....	19
Hình 10. Quá trình tính trọng số attention.....	20
Hình 11. Tính toán trọng số multi-head-attention	21
Hình 12 Hàm kích hoạt Gelu và đạo hàm	23
Hình 13 Hàm Relu.....	23
Hình 14 Bộ dữ liệu cuộc thi.....	32
Hình 15. Bộ dữ liệu được sinh bởi GPT có ngữ cảnh	34
Hình 16 Quá trình trả lời câu hỏi.....	39
Hình 17 Hình ảnh kết quả đào tạo deberta	43
Hình 18 Biểu đồ train_loss và val_loss	44
Hình 19 Biểu đồ đánh giá accuracy.....	44
Hình 20 Biểu đồ sự thay đổi của learning_rate	45
Hình 21. Kết quả quá trình nộp bài	46

MỞ ĐẦU

Ngôn ngữ tự nhiên là giao tiếp cơ bản của con người, không chỉ chứa đựng thông tin mà còn là phương tiện để chia sẻ kiến thức, ý định và cảm xúc. Trong thời đại số hóa hiện nay, sự gia tăng về dữ liệu văn bản và sự phát triển vượt bậc của máy móc đã mở ra những triển vọng mới cho việc hiểu và tương tác với ngôn ngữ tự nhiên, điều mà chúng ta thường gọi là Natural language Processing(NLP).

NLP là một lĩnh vực nghiên cứu tập trung vào khả năng của máy tính để hiểu, tạo ra và tương tác với ngôn ngữ tự nhiên một cách tự động và linh hoạt. Điều này bao gồm nhiệm vụ như dịch máy, tổng hợp giọng nói, phân loại văn bản và đặc biệt là trong trường hợp của đề tài, trả lời câu hỏi trắc nghiệm.

Trong môi trường ngôn ngữ tự nhiên, mọi đoạn văn bản và câu hỏi và phản hồi đều mang theo những đặc điểm phức tạp. Không chỉ về mặt ngữ nghĩa và còn về mặt ngữ pháp, ngữ cảnh, và thậm chí là sắc thái cảm xúc. NLP đặt ra thách thức lớn là làm thế nào máy tính có thể hiểu và phản ứng đúng với ngữ nghĩa này.

Với sự gia tăng đáng kể của dữ liệu và tiến bộ trong thuật toán học máy, NLP đã trở thành một trong những lĩnh vực nổi bật nhất trong cộng đồng nghiên cứu. Tính ứng dụng rộng lớn của NLP không chỉ giới hạn trong các dự án nghiên cứu mà còn lan rộng đến các sản phẩm hàng ngày như trợ lý ảo, công cụ dịch thuật thông minh và giải pháp trả lời câu hỏi tự động.

Chúng ta sẽ đi sâu và thách thức và triển vọng của NLP, đặc biệt là khi áp dụng nó và bài toán thú vị và thực tế- Trả lời câu hỏi trắc nghiệm.

Đặt vấn đề

Bài toán trả lời câu hỏi trắc nghiệm được đề xuất bởi cuộc thi của kaggle có tên “Kaggle-LLM science exam” với mục tiêu ứng dụng mô hình ngôn ngữ lớn(large language model) trả lời câu hỏi khoa học khó. Mục tiêu của cuộc thi lấy cảm hứng từ tập dữ liệu

openbookQA, cuộc thi này thách thức người chơi tham gia trả lời các câu hỏi khó dựa trên cơ sở khoa học được viết bởi mô hình ngôn ngữ lớn.

Bài toán không chỉ là một thách thức ngôn ngữ mà còn là một bài toán đa chiều đòi hỏi kết hợp giữa hiểu biết về văn bản, ngữ cảnh và khả năng xử lý thông tin. Đối mặt với ngôn ngữ tự nhiên đa dạng và phức tạp, chúng ta đặt ra một số vấn đề quan trọng cần giải quyết:

- Đa nghĩa và ngữ cảnh: Câu hỏi trắc nghiệm thường chứa từ ngữ đa nghĩa và yêu cầu sự hiểu biết rõ ràng về ngữ cảnh để đưa ra câu trả lời chính xác. Làm thế nào máy tính có thể hiểu được ngữ cảnh và xử lý những từ ngữ có nhiều ý nghĩa.
- Phức tạp với ngôn ngữ tự nhiên: Ngôn ngữ tự nhiên thường phức tạp, chứa đựng ngữ pháp đa dạng, từ vựng rộng lớn và cấu trúc phức tạp.
- Hiểu biết sâu sắc về văn bản: Để trả lời câu hỏi, máy tính cần hiểu biết sâu sắc về nội dung của văn bản và khả năng suy luận. Làm thế nào chúng ta có thể xây dựng mô hình có khả năng đọc và hiểu nội dung một cách linh hoạt?
- Một mô hình NLP hiệu quả cho bài toán trả lời câu hỏi trắc nghiệm cần đảm bảo sự linh hoạt trong việc xử lý các dạng câu hỏi khác nhau và đồng thời đạt được độ chính xác cao.

Mục tiêu nghiên cứu

Xây dựng được mô hình trả lời câu hỏi trắc nghiệm liên quan tới câu hỏi khoa học khó. Mô hình được thiết kế để :

- Hiểu và xử lý ngôn ngữ tự nhiên đa dạng : Phát triển khả năng của mô hình trong việc hiểu và xử lý ngôn ngữ tự nhiên đa dạng , bao gồm ngữ pháp, từ vựng và cấu trúc câu phức tạp. Mục tiêu là tạo ra một mô hình có thể hiểu và trả lời câu hỏi một cách chính xác.

- Xử lý đa nghĩa và ngữ cảnh: Xây dựng khả năng xử lý đa nghĩa và ngữ cảnh. Mô hình sẽ được huấn luyện để hiểu và đánh giá ngữ cảnh từ ngữ, giúp nó chọn lựa và đưa ra câu trả lời chính xác dựa trên ngữ cảnh.
- Tiến hành đánh giá độ chính xác của mô hình và submit kết quả cho cuộc thi. Đánh giá kết quả đạt được.

Ý nghĩa của bài toán

Nhiệm vụ trả lời câu hỏi trắc nghiệm trong mô hình học sâu có rất nhiều ứng dụng thực tế, chẳng hạn như hệ thống trả lời câu hỏi tự động, trợ lý ảo và nhiều ứng dụng khác trong lĩnh vực y tế, giáo dục, kỹ thuật... Mô hình học sâu ngày càng phát triển và hoàn thiện, giúp tối ưu hóa hiệu suất và tạo ra các ứng dụng tiện ích mạnh mẽ cho xã hội. Từ đó em quyết định chọn đề tài **“Xây dựng mô hình học sâu trong ngôn ngữ tự nhiên cho nhiệm vụ trả lời câu hỏi trắc nghiệm.”**

Phương pháp nghiên cứu

- Xây dựng dữ liệu: Để xây dựng mô hình NLP hiệu quả, em sử dụng một tập dữ liệu đa dạng chứa câu hỏi trắc nghiệm và câu trả lời tương ứng từ nhiều nguồn khác nhau về khoa học. Dữ liệu được làm sạch và tiền xử lý để loại bỏ nhiễu và đảm bảo tính nhất quán.
- Kiến trúc mô hình : Mô hình của tôi dựa trên kiến trúc Transformer, một trong những kiến trúc hiệu quả trong NLP. Tôi lựa chọn và tinh chỉnh các mode pre-train cho dữ liệu tiếng anh được chia sẻ từ huggingface và training thêm với bộ dữ liệu của mình.
- Huấn luyện và đánh giá: Mô hình được huấn luyện trên dữ liệu được chia thành tập huấn luyện và tập kiểm tra. Quá trình huấn luyện sử dụng thuật toán tối ưu hóa và được theo dõi bằng các độ đo hiệu suất như độ chính xác và giá trị mất mát trên tập kiểm tra. Để đánh giá hiệu suất, chúng tôi sử dụng các tập kiểm tra chứa các câu hỏi đa dạng và đánh giá kết quả dự đoán của mô hình so với câu trả lời thực tế. Các độ đo

như độ chính xác F1 score, recall, .. được sử dụng để đánh giá hiệu suất của mô hình trên các tác vụ cụ thể.

- Ứng dụng RAG, hay còn được biết đến là Retrieval-Augmented Generation, là một phương pháp trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP). Nhằm tìm kiếm các thông tin liên quan tới câu hỏi và câu trả lời từ đó cung cấp thêm ngữ cảnh cho câu hỏi và tăng độ chính xác của câu trả lời.
- Thử nghiệm và đánh giá: Tôi thực hiện một loạt các thử nghiệm để đánh giá hiệu suất của mô hình trên các dạng câu hỏi đặc biệt và so sánh với các mô hình khác. Kết quả của mỗi thử nghiệm được thảo luận và đánh giá chi tiết trong phần Kết Quả và Thảo Luận.

Phương pháp nghiên cứu này được thiết kế để đảm bảo sự toàn diện và đánh giá sâu sắc về khả năng của mô hình NLP trong việc giải quyết bài toán trả lời câu hỏi trắc nghiệm.

Chương 1: Giới thiệu bài toán, giới thiệu tổng quan

1.1. Giới thiệu tổng quan

1.1.1. Ngữ cảnh lịch sử

- Sự xuất hiện đầu tiên của Natural Language Processing(NLP)

Lĩnh vực Natural Language Processing(NLP) xuất hiện đầu tiên vào những năm 1950 khi các nhà nghiên cứu bắt đầu quan tâm đến khả năng của máy tính trong việc xử lý ngôn ngữ tự nhiên. Điều này đánh dấu một bước quan trọng trong lịch sử công nghệ, khi mà khả năng tương tác giữa con người và máy tính bắt đầu trở thành một ước mơ có thể biến thành hiện thực.

- Các tiến bộ quan trọng trong lịch sử

Những năm 1950 và 1960 là giai đoạn đánh dấu bởi những tiến bộ quan trọng trong lĩnh vực NLP. Các nhà nghiên cứu như Alan Turing đã đưa ra những ý tưởng về máy tính có thể mô phỏng khả năng thực hiện cuộc trò chuyện với con người. Tuy nhiên, trong những thập niên đầu tiên, tiến triển chủ yếu là ở mức độ lý thuyết và sự hiểu biết sâu sắc về ngôn ngữ tự nhiên vẫn là một thách thức.

- Sự bùng nổ của Big Data và Machine Learning

Đến những năm 2000, sự bùng nổ của dữ liệu lớn và sự phát triển của machine learning đã thúc đẩy sự tiến bộ mạnh mẽ trong lĩnh vực NLP. Các mô hình thống kê truyền thống đã được thay thế bằng những mô hình dựa trên dữ liệu và học máy, giúp máy tính hiểu biết và tương tác với mô hình ngôn ngữ tự nhiên một cách ngày càng tự nhiên.

- Mô hình Transformer và sự đột phá trong NLP

Năm 2018, Mô hình Transformer đã xuất hiện, đánh dấu một bước đột phá trong lĩnh vực xử lý ngôn ngữ tự nhiên(NLP). Được giới thiệu đầu tiên qua mô hình “Attention is All You Need” bởi Vaswani et al., Transformer đã mang lại một sự đột phá to lớn và thay đổi cách chúng ta hiểu và xử lý ngôn ngữ tự nhiên.

Kiến trúc Transformer

- Bộ Mã Hóa và Bộ giải mã : Transformer sử dụng cặp Bộ mã hóa và bộ giải mã để xử lý chuỗi. Cả hai đều sử dụng kiến trúc tự chú ý (self-attention), cho phép mô hình tập trung vào các phần quan trọng của câu.
- Kiến trúc Attention : Khả năng tự chú ý của Transformer giúp xác định mức độ quan trọng của từng từ trong câu, giống như các mô hình truyền thống chỉ dựa vào ngữ cảnh lân cận.

Ứng dụng của Transformer trong NLP

- Mô hình GPT(Generative Pre-trained Transformer): GPT là một trong những ứng dụng nổi bật của transformer. Nó sử dụng kiến trúc tự chú ý để sinh ra văn bản tự nhiên và đạt được hiệu suất ấn tượng trong nhiều tác vụ NLP.
- BERT (Bidirectional Encoder Representations from Transformers) : BERT là một mô hình Transformer biểu diễn một lựa chọn cách tiếp cận mới bằng cách sử dụng bối cảnh từ cả hai phía của mỗi từ (hai chiều), giúp cải thiện khả năng hiểu biết ngôn ngữ.

Lợi ích đối với NLP

- Độ phức tạp thấp và hiệu quả: Transformer giúp giải quyết vấn đề về vanishing gradient và exploding gradient, giảm độ phức tạp tính toán và tăng tốc độ huấn luyện so với các mô hình truyền thống.
- Đa nhiệm và học qua chuyển giao: Transformer có khả năng thực hiện nhiều tác vụ NLP mà không cần đào tạo lại hoàn toàn, tạo điều kiện thuận lợi cho học chuyển giao đa nhiệm.
- Mở rộng và ứng dụng rộng rãi : Mô hình Transformer đã trở thành một kiến trúc cơ bản, được sử dụng khác nhau như dịch máy, trả lời câu hỏi và phân loại văn bản.

Mô hình Transformer đánh dấu một bước tiến quan trọng, không chỉ trong NLP mà còn cả lĩnh vực machine learning, mở ra những tiềm năng mới và thách thức đối với các nhà nghiên cứu và những người làm công nghệ.

Trong thời đại số hóa, việc tự động trả lời câu hỏi trắc nghiệm bằng máy tính đã trở thành một khía cạnh quan trọng trong xử lý ngôn ngữ tự nhiên(NLP). Điều này không chỉ mang lại sự thuận tiện mà còn giúp tăng cường hiệu suất và tăng cường trải nghiệm của người dùng trong nhiều lĩnh vực, từ giáo dục đến công nghiệp.

1.2. Lý do lựa chọn đề tài

Với sự bùng nổ thông tin trực tuyến, việc tự động hóa quá trình trả lời câu hỏi trắc nghiệm trở nên ngày càng quan trọng. Đề tài này hướng tới việc phát triển hệ thống NLP có khả năng hiểu biết và xử lý câu hỏi trắc nghiệm, mang lại giải pháp hiệu quả và linh hoạt cho nhu cầu ngày càng tăng về học tập và kiểm tra trực tuyến.

1.2.1. Thách thức trong xử lý câu hỏi trắc nghiệm

Hệ thống trả lời câu hỏi trắc nghiệm đặt ra những thách thức đặc biệt trong việc hiểu biết ngôn ngữ tự nhiên, xử lý ngữ cảnh và đảm bảo độ chính xác cao. Đề tài được chọn để tìm hiểu và giải quyết những thách thức này, đồng thời đóng góp vào sự phát triển cần thiết của hệ thống thông minh trong lĩnh vực NLP.

1.2.2. Sự cần thiết cho giáo dục và doanh nghiệp

Sự tiện lợi và linh hoạt của hệ thống trả lời câu hỏi trắc nghiệm NLP không chỉ làm gia tăng hiệu suất học tập mà còn mang lại giá trị trong các ứng dụng doanh nghiệp, giúp tiết kiệm thời gian và nguồn lực.

1.3. Mục tiêu nghiên cứu

Mục tiêu chính của nghiên cứu là xây dựng và đánh giá một hệ thống trả lời câu hỏi trắc nghiệm dựa trên NLP, với khả năng hiểu biết ngôn ngữ tự nhiên (tiếng anh) và lựa chọn câu trả lời chính xác.

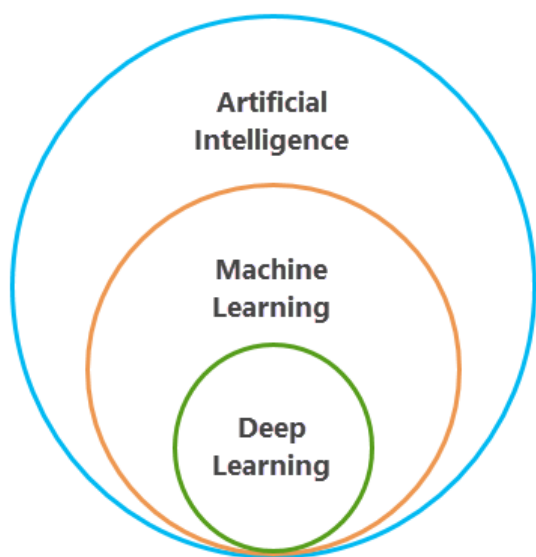
1.4. Phạm vi nghiên cứu

Nghiên cứu tập trung vào phát triển một mô hình NLP có khả năng xử lý câu hỏi trắc nghiệm trong các câu hỏi liên quan tới khoa học và cho ngôn ngữ tiếng anh. Với khả năng hiểu biết ngôn ngữ tự nhiên và cung cấp câu trả lời chính xác.

Chương 2: Các kỹ thuật sử dụng trong bài toán về NLP

Chương này tập trung thảo luận về các kỹ thuật quan trọng được sử dụng trong bài toán trả lời câu hỏi trắc nghiệm sử dụng Natural Language Processing (NLP). Bằng cách nghiên cứu và áp dụng những kỹ thuật này chúng ta có thể xây dựng những hệ thống có khả năng hiểu và xử lý ngôn ngữ tự nhiên để lựa chọn đáp án chính xác nhất cho câu hỏi trắc nghiệm. Cụ thể chương này sẽ chia thành các phần chính sau:

2.1. Deep learning



Hình 1 Hình ảnh học sâu trong trí tuệ nhân tạo

Deep learning(học sâu) là một phần của trí tuệ nhân tạo, nó tập trung vào việc xây dựng và huấn luyện các mô hình máy học sâu, có nghĩa là chúng có nhiều lớp nơ-ron. Deep learning giúp máy tính học từ dữ liệu phức tạp và tự tạo ra các biểu diễn phức tạp của thông tin.

Các mô hình deep learning thường lớn với hàng trăm hoặc thậm chí hàng nghìn lớp nơ-ron và triệu tham số. Để huấn luyện chúng, cần lượng lớn dữ liệu và tài nguyên tính toán, nhưng kết quả có thể là khả năng hiểu biết sâu sắc và khả năng thích ứng với dữ liệu mới.

Deep learning được xây dựng để mô phỏng khả năng tư duy của bộ não con người. Quy trình hoạt động cũng có một số đặc điểm sau:

Kiến Trúc Đa Lớp: Mạng nơ-ron được chia thành nhiều lớp, với càng nhiều lớp thì mạng càng "sâu". Mỗi lớp chứa các nút nơ-ron và các kết nối giữa chúng.

Trọng Số Kết Nối: Mỗi kết nối giữa các nút có một trọng số tương ứng, thể hiện mức độ quan trọng của kết nối đó đối với mạng. Trọng số càng cao, ảnh hưởng càng lớn.

Chức Năng Kích Hoạt: Mỗi nơ-ron có một chức năng kích hoạt, có trách nhiệm chuẩn hóa đầu ra từ nơ-ron. Chức năng này giúp định rõ sự kích thích và truyền thông tin trong mạng.

Dữ Liệu Đầu Vào và Đầu Ra: Dữ liệu được người dùng nhập vào mạng thần kinh và trải qua tất cả các lớp. Kết quả được sinh ra ở lớp đầu ra, nơi mà mô hình đưa ra dự đoán hoặc phân loại.

Quá Trình Đào Tạo: Trong quá trình đào tạo, mô hình cố gắng điều chỉnh các trọng số để tối ưu hóa hiệu suất. Điều này thường được thực hiện thông qua các thuật toán như lan truyền ngược, giúp mô hình học từ dữ liệu và cải thiện khả năng dự đoán.

Ưu Điểm:

Khả Năng Học Đặc Trưng Tự Động: Deep learning có khả năng tự học và trích xuất các đặc trưng phức tạp từ dữ liệu, giúp giảm thiểu sự phụ thuộc vào việc thiết kế đặc trưng thủ công.

Hiệu Suất Cao Trong Nhiều Lĩnh Vực: Trong nhiều ứng dụng như nhận diện hình ảnh, xử lý ngôn ngữ tự nhiên, và dự đoán chuỗi thời gian, deep learning đã đạt được hiệu suất vượt trội so với các phương pháp truyền thống.

Khả Năng Tự Điều Chỉnh (Adaptability): Mô hình deep learning có khả năng thích ứng với dữ liệu mới một cách tự động, giúp duy trì hiệu suất khi có thêm thông tin.

Đa Dạng và Linh Hoạt: Deep learning có thể được áp dụng trong nhiều lĩnh vực khác nhau, từ y tế đến tài chính, và có thể mô hình hóa nhiều loại dữ liệu khác nhau.

Nhược Điểm:

Yêu Cầu Dữ Liệu Lớn: Deep learning thường đòi hỏi lượng lớn dữ liệu để đạt hiệu suất tốt, điều này có thể là một thách thức trong những ứng dụng có ít dữ liệu.

Tính Phức Tạp Cao: Mô hình deep learning thường rất phức tạp và đòi hỏi tài nguyên tính toán lớn. Điều này có thể làm tăng chi phí triển khai và duy trì.

Nguy cơ Overfitting: Trong một số trường hợp, deep learning có thể dễ bị quá mức hóa mô hình (overfitting), đặc biệt là khi dữ liệu đào tạo không đủ.

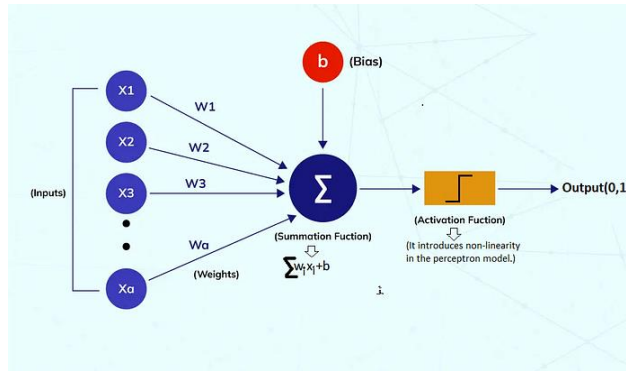
Deep learning được sử dụng rộng rãi trong nhiều lĩnh vực, từ nhận diện khuôn mặt và đối tượng trong hình ảnh, đến dịch máy tự động trong ngôn ngữ tự nhiên. Công nghệ này đang mang lại nhiều cơ hội mới và thách thức thú vị trong nghiên cứu và thực tế.

2.2. Neural network

Neural network (mạng nơ ron) là một phương pháp trong trí tuệ nhân tạo được áp dụng để dạy máy tính xử lý dữ liệu theo cách lấy cảm hứng từ bộ não con người. Bản chất mạng nơ ron là mô phỏng cách bộ não con người xử lý thông tin. Điều thú vị là nó không chỉ học từ dữ liệu, mà nó còn có khả năng tự điều chỉnh, tự cải thiện theo thời gian. Nếu bạn đặt cho nó một nhiệm vụ và nó làm sai, nó sẽ học từ sai lầm đó và thực hiện tốt hơn ở lần tiếp theo.

2.2.1. Perceptrons

Một mạng nơ-ron được cấu thành bởi các nơ-ron đơn lẻ được gọi là các perceptron.



Hình 2 Cấu tạo một Perceptron

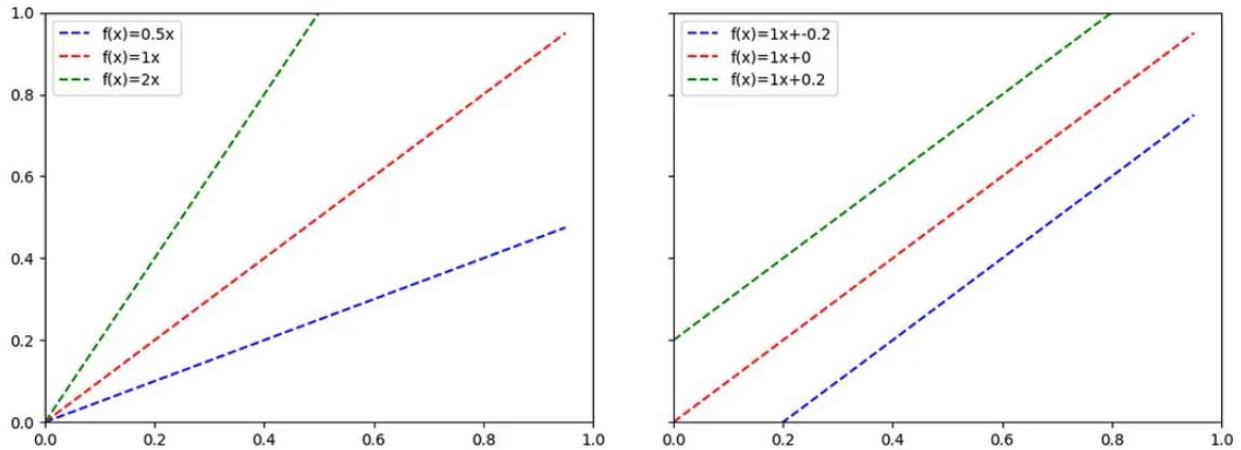
Các thành phần đơn lẻ của perceptron

- Inputs (Giá trị của một hoặc nhiều đầu vào)
- Weights and bias (trọng số và độ lệch)
- Weighted sum (Tổng trọng số)
- Activation function(Hàm kích hoạt)

Weights và bias là hai thành phần thiết yếu của mô hình perceptron. Đây là những tham số có thể học được và khi mạng được huấn luyện, cả hai tham số đều được điều chỉnh để đạt được giá trị và đầu ra mong muốn.

Weights (trọng số) được sử dụng để xác định tầm quan trọng của từng tính năng trong việc dự báo giá trị đầu ra. Các tính năng có giá trị gần bằng 0 được coi là có ít trọng số hoặc mức độ liên quan hơn. Những giá trị này ít quan trọng hơn trong quá trình dự đoán so với các đặc điểm có giá trị lớn hơn 0, được gọi là trọng số có giá trị cao hơn. Nếu trọng số dương thì nó có liên quan trực tiếp với giá trị mục tiêu, nếu nó âm, nó có mối quan hệ nghịch đảo với giá trị đích.

Độ lệch(bias) trong mô hình học máy được sử dụng để tạo ra sự linh động và điều chỉnh cho mô hình. Bias cho phép perceptron tính đến các tình huống trong đó các đầu vào không đủ để nắm bắt độ phức tạp của vấn đề. Nó cho phép perceptron tìm hiểu ranh giới quyết định tốt hơn bằng cách thay đổi chúng theo hướng mong muốn. Độ lệch cùng với các trọng số giúp mô hình có khả năng học được mối quan hệ phi tuyến giữa đầu vào và đầu ra, điều này là quan trọng để mô hình có thể xử lý các tình huống phức tạp. Độ lệch là một tham số quan trọng giúp mô hình nâng cao khả năng linh hoạt, thích ứng với dữ liệu và học được các mối quan hệ phức tạp trong quá trình huấn luyện.



Hình 3 Đồ thị của hàm với bias

Xét dạng chuẩn của hàm tuyến tính và bỏ qua hàm kích hoạt có công thức $f(x) = ax+b$. Độ lệch của chúng ta hiện tại là thành phần b . Nếu bỏ qua bias hay $b=0$ thì nhận thấy hàm luôn đi qua gốc tọa độ $[0,0]$. Nếu giữ nguyên a và thay đổi b thì các hàm mới sẽ luôn song song với nhau. Chúng ta sẽ nhận thấy rằng nếu không có thành phần bias sẽ mất đi tính linh hoạt của của hàm.

Hàm kích hoạt (activation function) là một thành phần quan trọng trong mạng neural, đặc biệt là trong các lớp ẩn của mô hình. “Phi tuyến tính hóa” Một trong những chức năng chính của hàm kích hoạt là giúp mô hình trở nên phi tuyến tính. Điều này cho phép mạng neural học được các biểu diễn phức tạp và phi tuyến tính của dữ liệu, giúp nó giải quyết các vấn đề phức tạp và không gian biểu diễn rộng lớn hơn. Non-linearity (Phi tuyến tính): Hàm kích hoạt giúp mô hình trở nên phi tuyến tính bằng cách thêm tính không tuyến tính vào đầu ra của các lớp ẩn. Điều này là quan trọng để mô hình có thể học được các mối quan hệ phức tạp giữa các đặc trưng đầu vào. Hàm kích hoạt giúp mô hình có khả năng học được biểu diễn đồng thời của đầu vào. Điều này có nghĩa là mô hình có thể xử lý và học được các tương tác phức tạp giữa các đặc trưng đầu vào.

Cách hoạt động của một perceptron

$$Output = f\left(\sum_{i=1}^n w_i x_i + b\right)$$

$F()$: là hàm kích hoạt

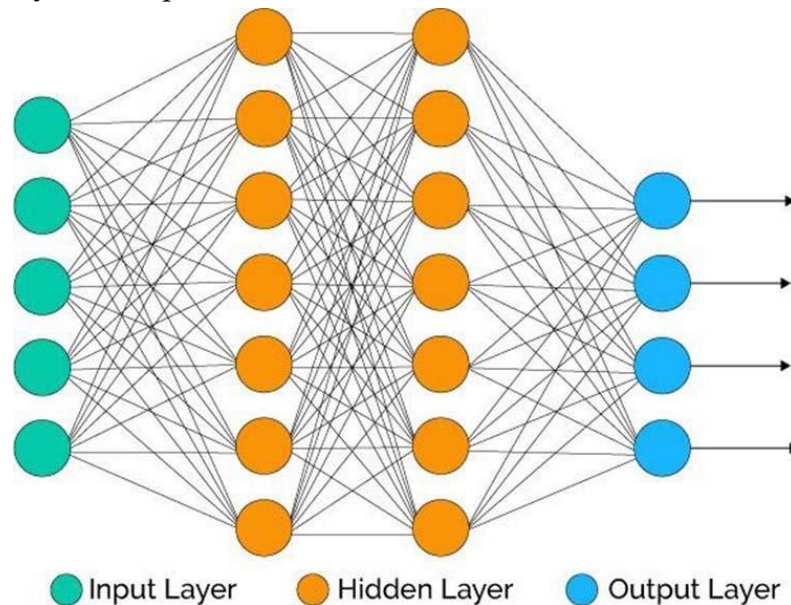
W_i : là trọng số tương ứng với mỗi đầu vào

X_i : là các giá trị đầu vào

b : là bias

Mô hình perceptron bắt đầu bằng phép nhân tất cả các giá trị đầu vào và trọng số của chúng, sau đó cộng các giá trị này lại với nhau để tạo ra tổng có trọng số. Sau đó, tổng có trọng số này được áp dụng cho hàm kích hoạt 'f' để thu được đầu ra mong muốn. Hàm kích hoạt này còn được gọi là hàm bước và được biểu thị bằng 'f'.

2.2.2. Multi-Layer Perceptron



Hình 4 kiến trúc mạng neural network

Neural network(mạng nơ ron) là sự kết hợp của những tầng perceptron hay còn gọi là perceptron đa tầng.

Tầng input layer (tầng vào): Tầng này nằm bên trái cùng của mạng, thể hiện cho các đầu vào của mạng.

Tầng output layer (tầng ra): Là tầng bên phải cùng và nó thể hiện cho những đầu ra của mạng.

Tầng hidden layer (tầng ẩn): Tầng này nằm giữa tầng vào và tầng ra, nó thể hiện cho quá trình suy luận logic của mạng.

Giống như mô hình perceptron một lớp, mô hình perceptron nhiều lớp cũng có cấu trúc mô hình tương tự nhưng có số lớp ẩn nhiều hơn.

Mô hình perceptron nhiều lớp còn được gọi là thuật toán Backpropagation, thực hiện theo hai giai đoạn như sau:

Giai đoạn chuyển tiếp: Các chức năng kích hoạt bắt đầu từ lớp đầu vào ở giai đoạn chuyển tiếp và kết thúc ở lớp đầu ra.

Giai đoạn lùi: Trong giai đoạn lùi, các giá trị trọng số và độ lệch được sửa đổi theo yêu cầu của mô hình. Trong giai đoạn này, sai số giữa đầu ra thực tế và yêu cầu bắt nguồn từ lớp đầu ra và kết thúc ở lớp đầu vào.

Do đó, mô hình perceptron nhiều lớp được coi là nhiều mạng nơ ron nhân tạo có nhiều lớp khác nhau trong đó hàm kích hoạt không duy trì tuyến tính, tương tự như mô hình perceptron một lớp. Thay vì tuyến tính, hàm kích hoạt có thể được thực thi dưới dạng sigmoid, TanH, ReLU, v.v. để triển khai.

2.3.Tokenization

Tokenization là quá trình chia nhỏ văn bản thành các đơn vị nhỏ hơn được gọi là tokens. Đây là một bước quan trọng trong tiền xử lý dữ liệu văn bản trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP). Tokenization giúp chúng ta chuyển đổi dữ liệu văn bản từ dạng chuỗi liên tục thành các đơn vị có ý nghĩa, tạo nên cơ sở cho việc phân tích và xử lý ngôn ngữ tự nhiên.

- Khái niệm Tokens
 - Đơn Vị Cơ Bản: Tokens là các đơn vị cơ bản trong NLP, đại diện cho các phần nhỏ nhất có ý nghĩa trong văn bản.
 - Đa Dạng: Tokens có thể là từ (word), từ phụ (sub-word), hoặc kí tự (character), phụ thuộc vào mục tiêu cụ thể và yêu cầu của ứng dụng.
- Mục tiêu Tokens
- Tạo Từ Vựng: Tokens được sử dụng để tạo từ vựng, một tập hợp các đơn vị duy nhất đại diện cho ngôn ngữ trong mô hình NLP.

- Mã Hóa: Mỗi token thường được biểu diễn bằng một số (ID), giúp máy tính hiểu và xử lý dữ liệu văn bản.
- Các phương pháp Tokenization

Word-Based Tokenization: Phổ biến nhất, chia văn bản thành các từ dựa trên dấu phân cách như dấu cách trắng.

Hạn chế chính của kỹ thuật tokenization dựa trên từ là sự tạo ra một kho từ vựng lớn và khối lượng dữ liệu ngôn ngữ lớn. Điều này dẫn đến mô hình NLP trở nên cồng kềnh và đòi hỏi tài nguyên tính toán đáng kể để xử lý. Một vấn đề khác là liên quan đến việc xử lý từ sai chính tả. Trong trường hợp một từ trong kho ngữ liệu bị viết sai, chẳng hạn như "knowledge" thành "knowldge," mô hình sẽ không nhận diện được từ này và thường sẽ gán cho nó một token OOV (Out-Of-Vocabulary)

Character-Based Tokenization: là quá trình chia một đoạn văn bản thành các đơn vị nhỏ nhất là các ký tự riêng lẻ. Thay vì sử dụng từ hoặc từ ngữ là đơn vị cơ bản như trong word-based tokenization, trong character-based tokenization, mỗi ký tự trong văn bản được coi là một token độc lập.

Ưu điểm của character-based tokenization bao gồm:

- Giảm độ phức tạp của từ vựng: Vì không cần xử lý và lưu trữ thông tin về từ ngữ, nên kích thước của từ vựng trở nên đơn giản hơn. Điều này có thể giúp giảm độ phức tạp của mô hình và giảm bộ nhớ cần thiết.
- Khả năng xử lý đa ngôn ngữ: Vì ký tự thường có sẵn trong nhiều ngôn ngữ khác nhau, character-based tokenization có thể áp dụng cho văn bản đa ngôn ngữ mà không cần điều chỉnh từ vựng.

Nhược điểm :

- Tăng độ dài chuỗi: Do mỗi ký tự được coi là một token, độ dài của chuỗi sẽ tăng lên đáng kể so với word-based tokenization. -Điều này có thể tạo ra các chuỗi rất dài và ảnh hưởng đến hiệu suất của mô hình.
- Mất thông tin ngữ cảnh từ từ ngữ: Việc không sử dụng thông tin từ ngữ có thể làm mất đi sự hiểu biết về ngữ cảnh của một từ trong câu.

Character-based tokenization thường được sử dụng trong các trường hợp đặc biệt, như khi xử lý văn bản tiếng Trung, nơi từ ngữ thường được biểu diễn bằng các ký tự. Tùy thuộc vào bài toán và loại dữ liệu, người ta sẽ chọn phương pháp tokenization phù hợp để đạt được kết quả tốt nhất

Subword-Based Tokenization: là quá trình chia văn bản thành các đơn vị nhỏ hơn từ, thường là các từ phụ (subwords) nhỏ hoặc các phần tử ngôn ngữ nhỏ hơn, giữa giữa từ và ký tự.

Một số ưu điểm của subword-based tokenization bao gồm:

- Giảm kích thước từ vựng: Subword-based tokenization giúp giảm kích thước của từ vựng so với word-based tokenization. Thay vì biểu diễn từng từ là một token, mô hình sử dụng các đơn vị nhỏ hơn như từ phụ, giúp giảm số lượng token cần xử lý.
- Xử lý từ mới linh hoạt: Subword tokenization cho phép mô hình xử lý từ mới hoặc từ không xuất hiện trong từ vựng một cách linh hoạt hơn. Điều này đặc biệt hữu ích khi mô hình phải đối mặt với ngôn ngữ mới, thuật ngữ chuyên ngành, hoặc từ ngữ không phổ biến.
- Tích hợp sự linh hoạt của character-based và word-based tokenization: Subword-based tokenization nằm giữa character-based và word-based tokenization, giúp giữ lại một số thông tin ngữ cảnh của từ trong khi vẫn giảm kích thước của từ vựng.

Một số thuật toán nổi tiếng cho subword-based tokenization bao gồm Byte Pair Encoding (BPE), SentencePiece, và WordPiece. Các thuật toán này thường sử dụng các phương

pháp như tìm kiếm các chuỗi phổ biến nhất và thay thế chúng bằng một token mới để tạo ra các đơn vị nhỏ hơn.

Hầu hết các mô hình tiếng Anh đều sử dụng các dạng thuật toán của mã hóa từ phụ, trong đó, phổ biến là WordPeces được sử dụng bởi BERT và DistilBERT, Unigram của XLNet và ALBERT, và Byte-Pair Encoding của GPT-2 và RoBERTa.

Tổng quát, subword-based tokenization thường được ưa chuộng trong các ứng dụng xử lý ngôn ngữ tự nhiên, đặc biệt là khi mô hình cần xử lý nhiều loại ngôn ngữ hoặc phải làm việc với từ vựng đa dạng.

Byte Pair Encoding (BPE) là một phương pháp subword-based tokenization được sử dụng để chia nhỏ các từ thành các subwords nhỏ hơn. Phương pháp này giúp giảm kích thước từ vựng, đồng thời tăng khả năng xử lý từ mới và từ ngữ đa dạng.

Cách thức hoạt động của BPE như sau:

Bước 1: Tính tần suất xuất hiện của các cặp byte (byte pair): Ban đầu, mỗi từ được chia thành các byte (ký tự hoặc các đơn vị nhỏ hơn). Sau đó, tần suất xuất hiện của các cặp byte liên tiếp được đếm trong toàn bộ tập dữ liệu.

Bước 2: Kết hợp các cặp byte xuất hiện nhiều nhất: Cặp byte xuất hiện nhiều nhất được kết hợp lại để tạo ra một subword mới. Quá trình này được lặp lại một số lần, hoặc cho đến khi đạt đến một số lượng subwords được xác định trước.

Bước 3: Tạo từ vựng mới: Các subword mới được thêm vào từ vựng, và quá trình lặp lại để tạo ra các subwords tiếp theo cho đến khi đạt được số lượng subwords mong muốn.

Đối với mỗi từ, BPE thường tạo ra một chuỗi các subwords nhỏ hơn, từ đơn vị ký tự đến các subword độc lập. Các từ phổ biến sẽ được giữ lại trong dạng nguyên vẹn, trong khi các từ hiếm hoặc từ mới có thể được phân tách thành các subwords để giúp mô hình hiểu biết tốt hơn.

Các thư viện và công cụ:

NLTK, spaCy, Keras, Gensim: Cung cấp các công cụ và thư viện hỗ trợ quá trình tokenization trong Python.

2.4. Word Embedding và Vectorization

Word Embedding là một kỹ thuật biểu diễn từng từ trong ngôn ngữ bằng các vector số thực trong không gian nhiều chiều. Mục tiêu của Word Embedding là ánh xạ các từ có ý nghĩa tương tự gần nhau trong không gian vector. Điều này giúp mô hình hiểu được mối quan hệ ngữ nghĩa giữa các từ.

Các phương pháp phổ biến của Word Embedding bao gồm:

- Word2Vec: Sử dụng mô hình dự đoán ngữ cảnh (context) của từ để học các vector biểu diễn.
- GloVe (Global Vectors for Word Representation): Kỹ thuật này tận dụng thông tin thống kê toàn bộ tập dữ liệu để học các vector biểu diễn.
- FastText: Mở rộng Word2Vec bằng cách biểu diễn từng từ dưới dạng tổ hợp các n-gram của ký tự.

Vectorization: Vectorization là quá trình biến đổi dữ liệu từ định dạng không phải vector thành dạng vector, thường được sử dụng trong machine learning và xử lý ngôn ngữ tự nhiên. Trong ngữ cảnh xử lý ngôn ngữ tự nhiên, vectorization có thể áp dụng cho văn bản để biểu diễn từ và văn bản dưới dạng vector số.

Trong kiến trúc Transformer, Word Embedding và Vectorization vẫn đóng một vai trò quan trọng trong việc biểu diễn và xử lý ngôn ngữ tự nhiên, nhưng có những sự khác biệt so với các mô hình truyền thống.

Trong Transformer, word embedding được thực hiện thông qua một lớp embedding đặc biệt, thường được gọi là Embedding Layer. Mỗi từ trong câu được ánh xạ sang một vector số thực trong không gian nhiều chiều thông qua embedding layer. Điều đặc biệt ở Transformer là các vector embedding này không chỉ biểu diễn từ riêng lẻ mà còn bao gồm thông tin ngữ cảnh của từ đó trong câu.

2.5.Pre-trained model

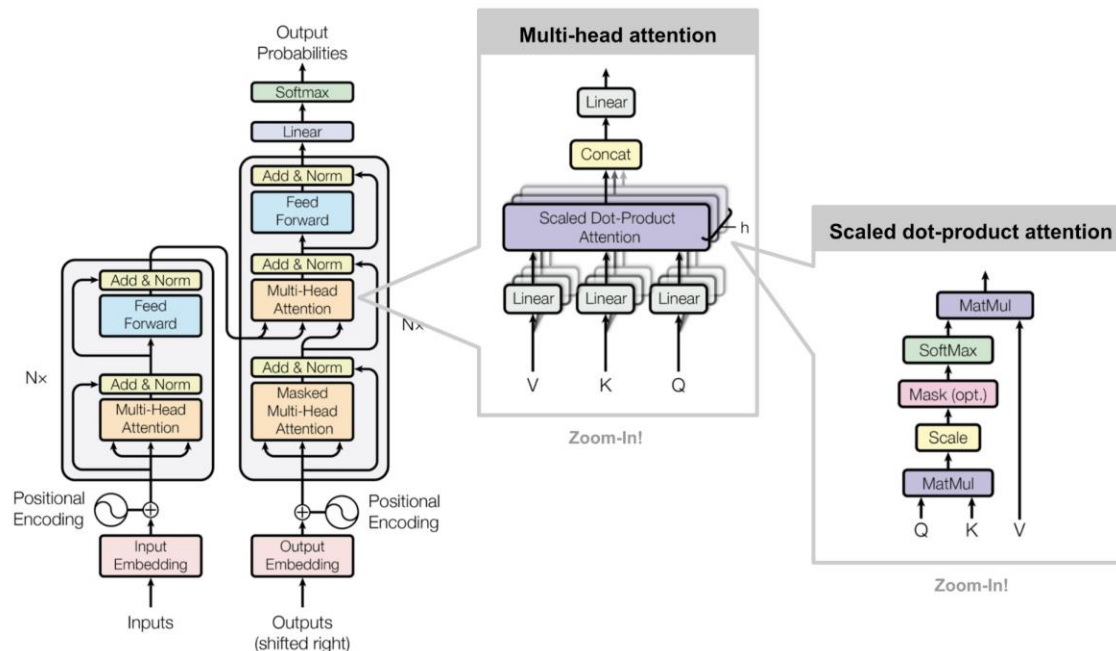
Pre-trained models là một loại mô hình máy học đã trải qua quá trình huấn luyện trước đó trên một lượng lớn dữ liệu. Trong ngữ cảnh của xử lý ngôn ngữ tự nhiên (NLP), mô hình đã được tiền huấn luyện thường là các mô hình ngôn ngữ có khả năng hiểu và sản xuất văn bản.

Mô hình đã được tiền huấn luyện thường được đào tạo trên các tập dữ liệu lớn, đa dạng của ngôn ngữ tự nhiên. Quá trình tiền huấn luyện giúp mô hình nắm bắt thông tin ngôn ngữ thông qua việc dự đoán từ tiếp theo trong một câu, điều này giúp nó học được cú pháp, ý nghĩa, và mối quan hệ giữa các từ.

Một số pre-trained model nổi tiếng trong lĩnh vực NLP bao gồm BERT (Bidirectional Encoder Representations from Transformers), GPT (Generative Pre-trained Transformer), và RoBERTa (Robustly optimized BERT approach). Các mô hình này đã đạt được những thành tựu ấn tượng trong nhiều nhiệm vụ NLP, như phân loại văn bản, dịch ngôn ngữ, và sinh văn bản tự động.

Một lợi ích lớn của việc sử dụng mô hình đã được tiền huấn luyện là khả năng chuyển giao (transfer learning). Bạn có thể sử dụng một mô hình đã được tiền huấn luyện trên một tập dữ liệu lớn để giúp cải thiện hiệu suất của mô hình cho một nhiệm vụ cụ thể mà không cần phải đào tạo lại từ đầu trên một tập dữ liệu nhỏ. Điều này giúp tiết kiệm thời gian và nguồn lực đào tạo, đặc biệt là khi tài nguyên hạn chế.

2.6. Transformer-Encoder



Hình 5 Kiến trúc transformer

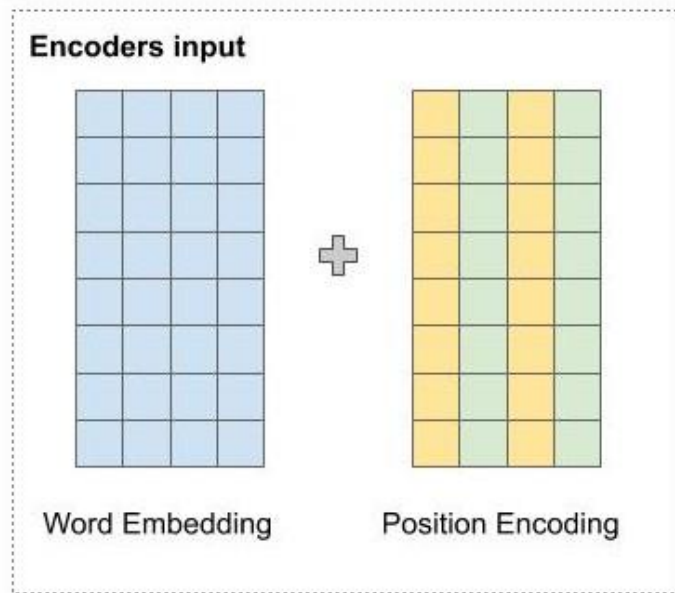
2.6.1. Embedding layer with Position Encoding

Embedding Layer chịu trách nhiệm ánh xạ các từ hoặc subwords thành các vector số thực trong không gian nhiều chiều. Mỗi từ trong câu được biểu diễn dưới dạng một vector embedding. Embedding Layer giúp mô hình hiểu biểu diễn ngữ nghĩa của từ và cách chúng tương tác trong không gian vector. Cụ thể, với từ vựng lớn, mỗi từ được ánh xạ vào một vector duy nhất trong không gian embedding. Những vector này sẽ được cập nhật và được sử dụng như làm đầu vào cho mô hình Transformer trong các tầng sau đó.

Một đặc điểm của Transformer là nó không duy trì thông tin về thứ tự từ trong câu như các mô hình RNN hoặc LSTM truyền thống. Do đó, cần thêm Positional Encoding để giúp mô hình hiểu vị trí của từ trong câu. Positional Encoding thêm thông tin về vị trí tuyến tính của từ vào vector embedding của nó. Phương pháp phổ biến là sử dụng hàm sin và cos để tạo ra các giá trị positional encoding có tính chất liên tục và dễ học.

Phương pháp đề xuất sinusoidal position encoding

Phương pháp được tác giả đề xuất: Vị trí của các từ được mã hóa bằng một vector có kích thước bằng word embedding và được cộng trực tiếp vào word embedding.



Hình 6 Encoders input

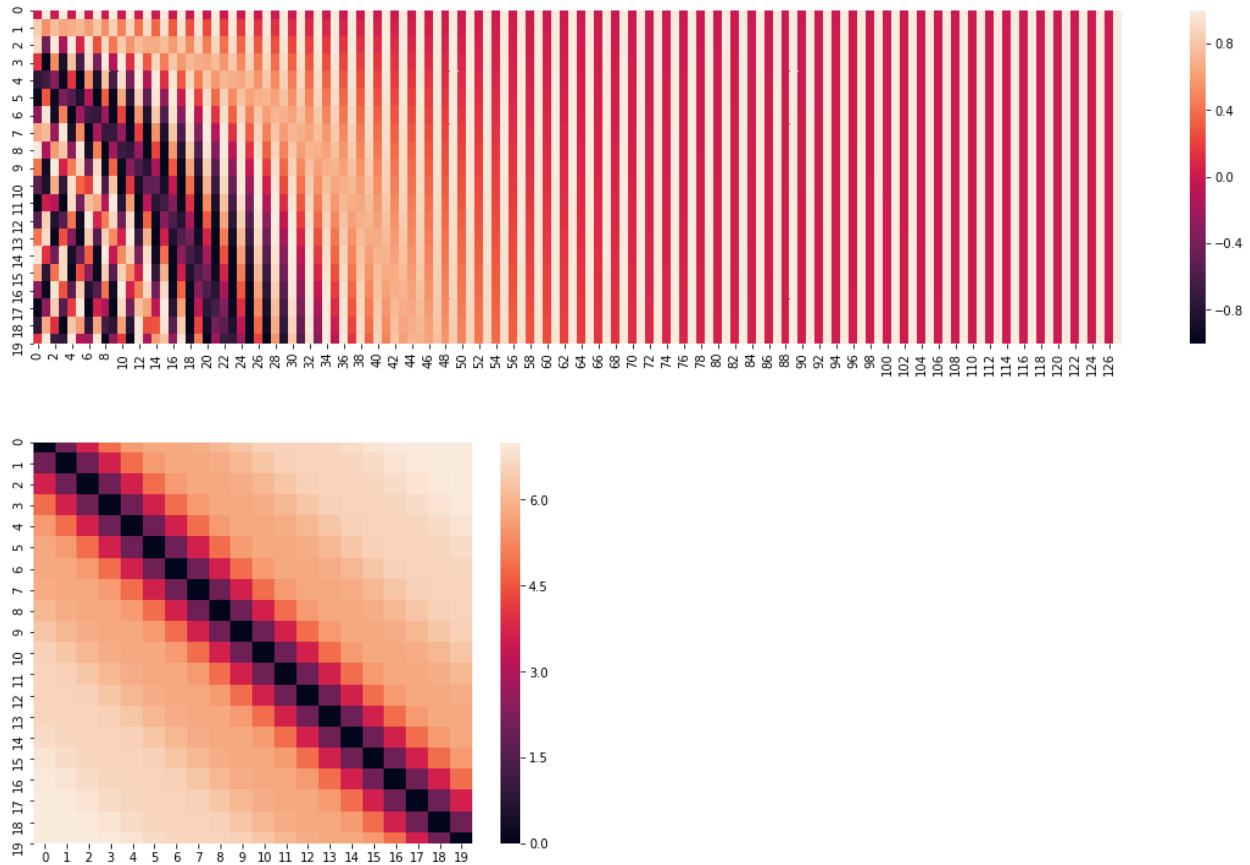
Sử dụng hàm sin để tính giá trị cho vị trí chẵn và hàm cos để tính giá trị cho vị trí lẻ.

$$p_t^i = f(t)^i = \begin{cases} \sin(\omega_k * t) & \text{if } i = 2k \\ \cos(\omega_k * t) & \text{if } i = 2k + 1 \end{cases}$$

Trong đó

$$\omega_k = \frac{1}{10000^{2k/d}}$$

Trong công thức được đề xuất, thấy rằng hàm sin và cos có dạng đồ thị tần số và tần số này giảm dần ở các chiều lớn dần. Các bạn xem hình dưới, ở chiều 0, giá trị này thay đổi liên tục tương ứng với màu sắc thay đổi liên tục và tần số thay đổi này giảm dần ở các chiều lớn hơn.

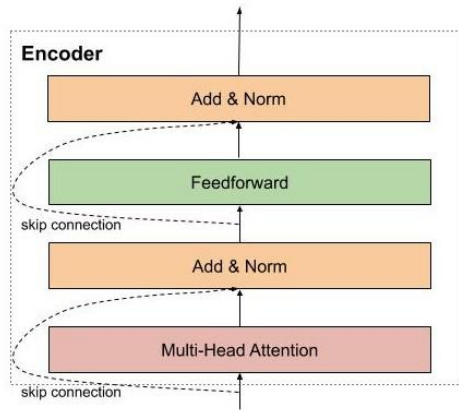


Hình 7 Sự thay đổi về tần số

Chúng ta có thể thấy các vector biểu diễn thể hiện được tính chất khoảng cách giữa 2 từ. 2 từ cách nhau càng xa thì khoảng cách càng lớn hơn.

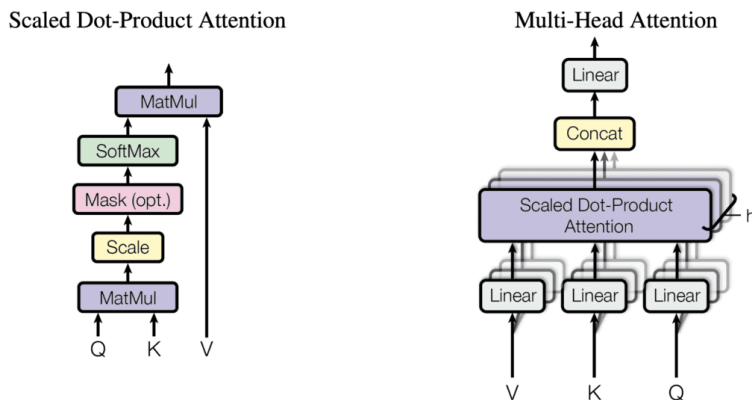
2.6.2. Encoder

Encoder của mô hình transformer có thể bao gồm nhiều encoder layer tương tự nhau. Mỗi encoder layer của transformer lại bao gồm 2 thành phần chính là multi head attention và feedforward network, ngoài ra còn có cả skip connection và normalization layer.



Hình 8. Cấu tạo lớp encoder

Encoder đầu tiên sẽ nhận ma trận biểu diễn của các từ đã được cộng với thông tin vị trí thông qua positional encoding. Sau đó, ma trận này sẽ được xử lý bởi Multi Head Attention.



Hình 9. Attention và multi-head-attention

Các thành phần chính được sử dụng bởi transformer:

- q và k biểu thị vector kích thước d_k , chứa các truy vấn và khóa tương ứng.
- d_k là chiều của vector key
- v biểu thị một vector thứ nguyên d_v chứa các giá trị
- Q, K, V biểu thị các ma trận theo bộ truy vấn(query), khóa (key), giá trị (value)
- W^Q, W^K, W^V biểu thị các ma trận chiếu được sử dụng để tạo các biểu diễn không gian con khác nhau của ma trận truy vấn(query), khóa (key), giá trị (value)

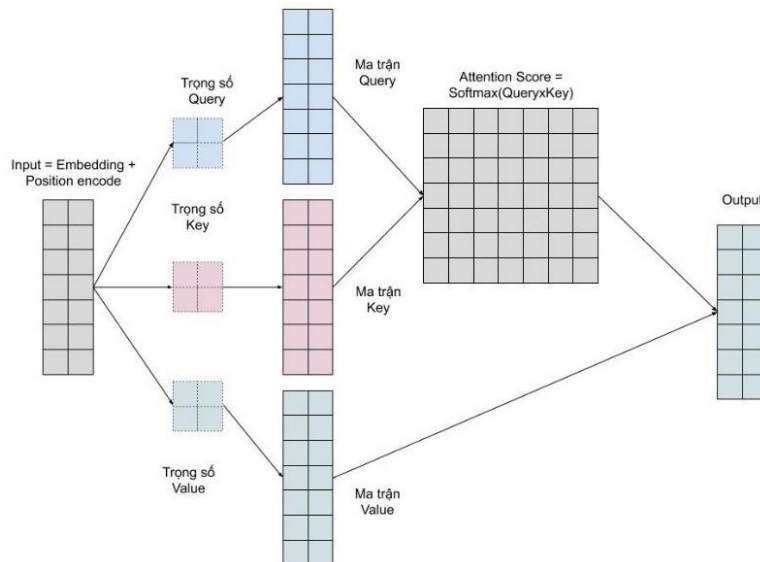
- W^O Biểu thị ma trận chiếu cho đầu ra nhiều đầu.

Scaled Dot-Product Attention

Scaled Dot-Product Attention là một phương pháp attention trong kiến trúc Transformer, được sử dụng để tính trọng số attention giữa các cặp từ trong một chuỗi văn bản. Phương pháp này giúp mô hình tập trung vào các phần quan trọng của chuỗi và hiểu mối quan hệ giữa các từ.

Cách hoạt động của scaled Dot-Product Attention

Tính trọng số attention



Hình 10. Quá trình tính trọng số attention

$$Attention(Q, K, V) = Softmax\left(\frac{Q \cdot K^t}{\sqrt{d_k}}\right) \cdot V$$

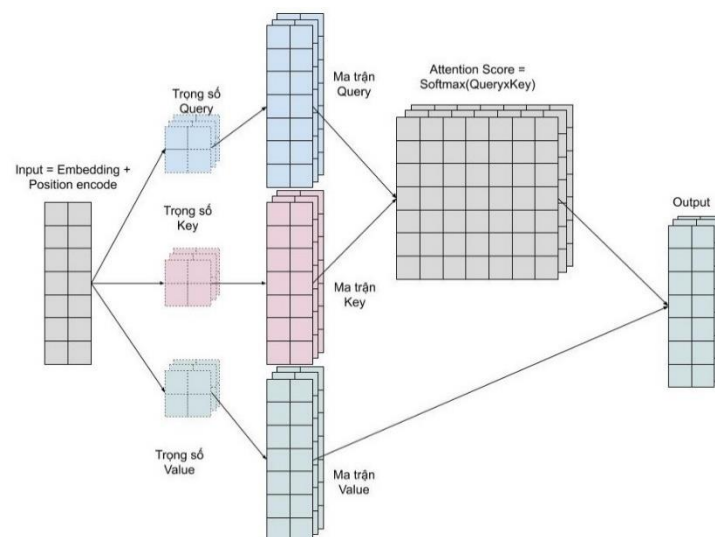
Bằng cách nhân 2 ma trận Q và K^t từ đó có được ma liên kết giữa mỗi từ với các từ còn lại. Tuy nhiên việc nhân ma trận có xu hướng tăng cường độ lớn. Có xu hướng làm cho hàm softmax trả về những giá trị rất nhỏ (Hàm softmax có xu hướng tăng cường giá trị lớn và giảm nhẹ giá trị nhỏ).

Để tránh tình trạng này chúng ta sử dụng scale kết quả dot-product với giá trị: $\sqrt{d_k}$

Softmax được sử dụng trong cơ chế Attention để chuyển đổi các điểm tương quan (tính bằng dot product) thành các trọng số attention có tổng bằng 1. Hàm softmax giúp tạo ra một phân phối xác suất, trong đó giá trị của mỗi phần tử trong phân phối là một số thực trong khoảng (0, 1) và tổng của chúng là 1. Từ đó giúp xác định mức độ quan trọng của các từ trong chuỗi, làm cho trọng số attention linh hoạt và tập trung vào phần quan trọng của chuỗi. Việc sử dụng Softmax trong Attention Mechanism giúp chúng ta biến đổi các điểm tương quan thành các trọng số attention theo cách có tính xác suất và có lợi ích trong việc tối ưu hóa và giảm vấn đề vanishing/exploding gradient

- Multi-head-attention

Chúng ta muốn mô hình có thể học nhiều kiểu mối quan hệ giữa các từ với nhau. Với mỗi self-attention, chúng ta học được một kiểu pattern, do đó để có thể mở rộng khả năng này, chúng ta đơn giản là thêm nhiều self-attention. Tức là chúng ta cần nhiều ma trận query, key, value mà thôi. Giờ đây ma trận trọng số key, query, value sẽ có thêm 1 chiều depth nữa.



Hình 11. Tính toán trọng số multi-head-attention

Mỗi đầu attention có thể học mối quan hệ và ngữ cảnh khác nhau, giúp mô hình học được sự phức tạp và đa dạng của dữ liệu ngôn ngữ. Các đầu attention không hoạt động độc lập

hoàn toàn, mà chúng còn chia sẻ thông tin thông qua việc concatenate và tương tác, giúp tăng cường khả năng tổng hợp thông tin của mô hình.

Multi-Head Attention là một phần quan trọng của Transformer, giúp mô hình hiểu biểu diễn ngôn ngữ tự nhiên và mối quan hệ ngữ cảnh giữa các từ trong câu.

- Residuals Connectin và Normalization layer

Residual Connections là một thành phần quan trọng trong các mô hình sâu (deep neural networks) như ResNet. Ý tưởng chính của Residual Connections là thêm đường tắt (shortcut) trực tiếp từ đầu vào tầng này đến đầu ra của tầng khác, qua cả một hàm ánh xạ. Cụ thể, nếu x là đầu vào của tầng, và $F(x)$ là ánh xạ học được bởi tầng, thì đầu ra của tầng đó là

$$y = F(x) + x$$

Residual connections giúp giảm vấn đề gradient vanishing/Exploding trong quá trình lan truyền ngược. Nếu tầng rồi được ánh xạ gần giống với một hàm đồng nhất, đường tắt sẽ đảm bảo được thông tin đầu vào được truyền qua mà không bị biến đổi quá mức.

Normalization Layer thường được sử dụng để giúp cải thiện hiệu suất và tốc độ học của mô hình. Normalization Layers giúp kiểm soát độ lớn của giá trị đầu ra từ mỗi tầng, giúp mô hình học được hiệu quả hơn và giảm vấn đề vanishing/exploding gradient.

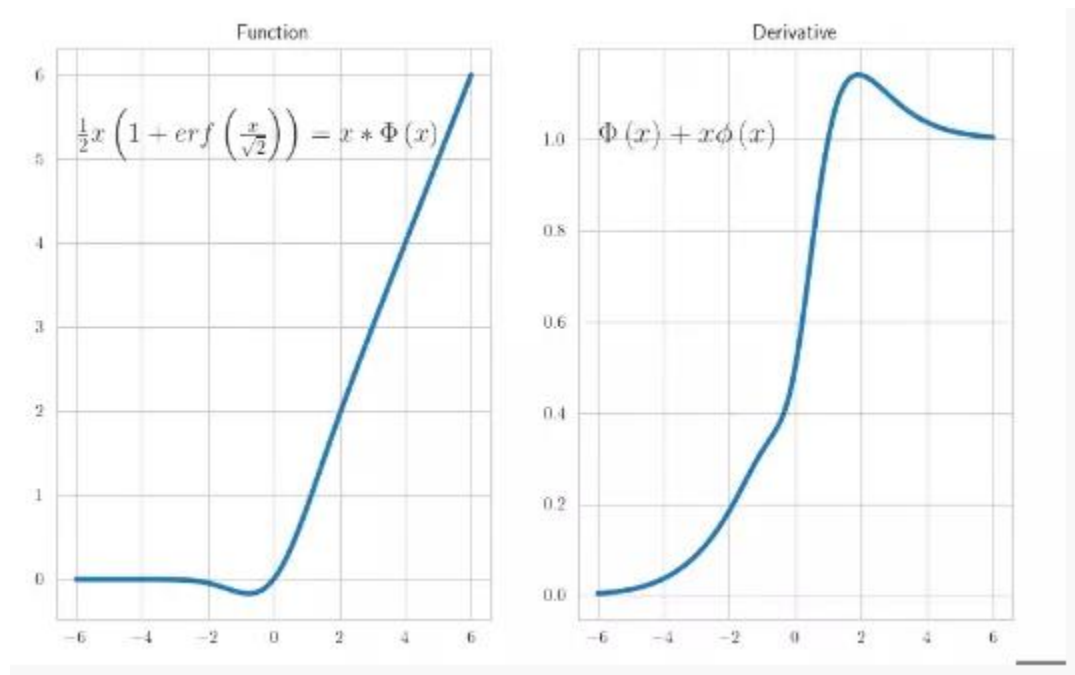
Trong kiến trúc của mô hình transformer, residuals connection và normalization layer được sử dụng nhiều lần. 2 kỹ thuật giúp cho mô hình huấn luyện nhanh hội tụ hơn và tránh mất mát thông tin trong quá trình huấn luyện mô hình, ví dụ như là thông tin của vị trí các từ được mã hóa.

2.7.Activation GELU

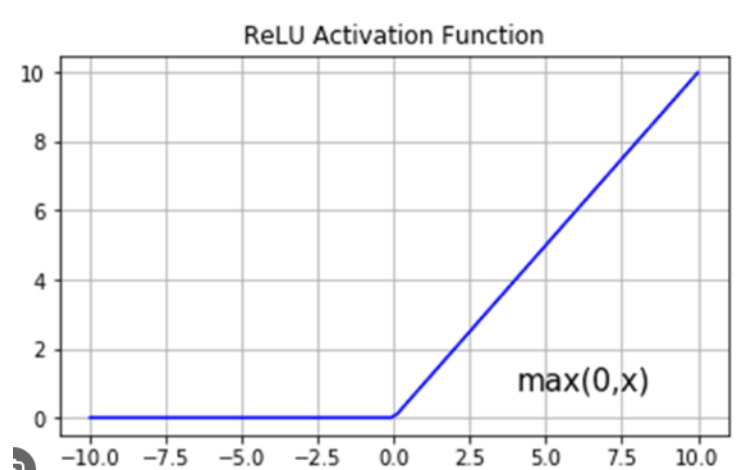
$$GELU_{(x)} = 0,5x \left(1 + \tanh \left(\sqrt{\frac{2}{\pi}} x + 0,044715x^3 \right) \right)$$

Transformer hiện đang rộng rãi sử dụng activation function GELU thay vì các hàm tanh hoặc sigmoid. Tác giả Dan Hendrycks và Kevin Gimpel đã mô tả trong bài báo của họ rằng sự hiệu quả của GELU đến từ việc kết hợp đầu vào x và phân phối Gaussian tiêu chuẩn. Hàm sigmoid, mặc dù phổ biến, thường gặp vấn đề "vanishing gradient" trong

mạng thần kinh sâu do giới hạn đầu ra và khả năng bão hòa. Ngược lại, các hàm kích hoạt như ReLU không có giới hạn này và thường hoạt động tốt hơn. Tuy nhiên, tác giả cho rằng GELU của họ vượt trội nhiều hơn, đặc biệt trong việc sửa đổi thích ứng dropout dựa trên khả năng phân phối xác suất.



Hình 12 Hàm kích hoạt Gelu và đạo hàm



Hình 13 Hàm Relu

Chúng ta nhận thấy rằng khi đạo hàm của hàm Gelu vượt qua giá trị 0 và 1, nó có thể là số âm hoặc dương, nhưng dù thế, tất cả giá trị này đều nằm trong khoảng từ 0 đến 1. Điều này cho thấy activation function này có khả năng kết hợp tính linear và tính non-linear, so

với một số activation khác. Đặc biệt, đường cong của nó trở nên mượt mà hơn, khắc phục vấn đề mà ReLU gặp phải khi bị bó cứng tại giá trị 0.

Activation function GELU không giống như ReLU, không bị triệt tiêu gradient do giá trị không bị giới hạn như sigmoid hoặc tanh. Điều này giúp các mô hình transformer ngày nay không bị vấn đề triệt tiêu gradient, mặc dù chúng có số lớp mạng sâu và được đào tạo sâu đặc. Điều này giúp chúng hội tụ một cách hiệu quả trong quá trình đào tạo.

2.7. Loss function Negative Log Likelihood Loss(NLLLoss)

Đối với một ví dụ huấn luyện duy nhất, giả sử xác suất dự đoán cho mỗi lớp là p_1, p_2, \dots, p_k , với k là số lớp. Nhãn lớp thực tế được ký hiệu là y , đó là một vector one-hot (một vector với tất cả các phần tử bằng không ngoại trừ chỉ mục tương ứng với lớp đúng, được đặt bằng 1).

Negative Log Likelihood Loss được tính như sau:

$$NLLLoss = -\log(P_y)$$

NLLLoss (Negative Log Likelihood Loss) là một hàm mất mát thường được sử dụng trong machine learning, đặc biệt là trong các bài toán phân loại. Hàm này thường được kết hợp với softmax activation ở lớp đầu ra của mạng neural cho các bài toán phân loại nhiều lớp.

Hàm mất mát Negative Log Likelihood Loss xuất phát từ khái niệm của ước lượng độ tin cậy tối đa. Trong bối cảnh phân loại, mục tiêu là tối đa hóa khả năng xuất hiện của lớp đúng dựa trên dữ liệu đầu vào. Log likelihood được chuyển thành negative để chuyển vấn đề tối đa thành vấn đề tối thiểu, phù hợp với các thuật toán tối ưu hóa.

Trong công thức trên p_y là xác suất dự đoán của lớp đúng. Công thức này trừng phạt mô hình nhiều hơn nếu xác suất dự đoán cho lớp đúng là thấp. Hàm mất mát được giảm thiểu khi xác suất dự đoán cho lớp đúng gần 1.

2.8. Tính toán độ tương đồng cosin, L2

Độ tương đồng của 2 vector có thể được tính toán bằng 2 độ đo: Cosine Similarity và Euclidean distance.

- Tính toán độ tương đồng Cosine

Độ tương đồng cosine giữa hai vector A và B được tính bằng công thức:

$$\text{Cosine Similarity}(A, B) = \frac{A \cdot B}{\|A\| \cdot \|B\|}$$

$$\|A\| = \sqrt{A_1^2 + A_2^2 + \dots + A_n^2}$$

Trong đó:

· là phép nhân dot product giữa A và B.

$\|A\|$ là độ dài (norm) của vector A.

$\|B\|$ là độ dài (norm) của vector B.

- Tính toán Độ Tương Đồng L2 (Euclidean)

Độ tương đồng L2 (Euclidean distance) giữa hai vector A và B được tính bằng công thức:

$$L_2(A, B) = \frac{1}{1 + \|A - B\|^2}$$

Trong cả hai trường hợp, giá trị càng gần 1 thì hai vector càng giống nhau, và giá trị càng gần 0 thì hai vector càng khác nhau.

2.9. Phương pháp đánh giá

2.9.1. Phương pháp đánh giá của cuộc thi

Đánh giá các đề xuất được thực hiện dựa trên chỉ số Trung bình Độ chính xác Trung bình @ 3 (MAP@3):

$$\text{Map@3} = \frac{1}{U} \sum_{u=1}^U \sum_{k=1}^{\min(n,3)} P(k) * \text{Rel}(k)$$

Trong đó:

U là số câu hỏi trong tập kiểm tra.

$P(k)$ là độ chính xác tại điểm cắt k . $(1/k)$

n là số dự đoán mỗi câu hỏi.

$rel(k)$ là một hàm chỉ số bằng 1 nếu mục ở vị trí k là nhãn liên quan (đúng), và bằng 0 nếu ngược lại.

Khi đã tính điểm cho một nhãn đúng cho một câu hỏi cụ thể trong tập kiểm tra, nhãn đó sẽ không còn được xem xét là liên quan cho câu hỏi đó nữa, và các dự đoán bổ sung của nhãn đó sẽ được bỏ qua trong tính toán.

Ví dụ:

Đầu ra top 3 đáp án đúng nhất được sắp xếp theo xác suất lớn đến bé cho câu hỏi là [B,A,C] . Đáp án đúng cho câu hỏi là A. Điểm số cho câu trả lời này được tính bằng:

$$\text{Score} = 1/1 * 0 + 1/2 * 1 + 1/3 * 0 = 0.5$$

2.9.2. Accuracy

Đây là độ đo của bài toán phân loại đơn giản nhất, tính toán bằng cách lấy tổng số câu trả lời đúng chia cho tổng số câu hỏi. Từ đó biết phần trăm số câu hỏi đúng và sai trên toàn bộ dữ liệu.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Nhược điểm của cách đánh giá này là chỉ cho ta biết được bao nhiêu phần trăm lượng dữ liệu được phân loại đúng mà không chỉ ra được cụ thể mỗi loại được phân loại như thế nào, lớp nào được phân loại đúng nhiều nhất hay dữ liệu của lớp nào thường bị phân loại nhầm nhất vào các lớp khác.

Chương 3: Ứng dụng cho nhiệm vụ trả lời câu hỏi trắc nghiệm

3.1. Giới thiệu

Nhiệm vụ trả lời câu hỏi trắc nghiệm là một trong những thách thức quan trọng của lĩnh vực xử lý ngôn ngữ tự nhiên (NLP), nơi các mô hình máy học được huấn luyện để đọc và hiểu nội dung văn bản, sau đó tự động trả lời các câu hỏi theo định dạng trắc nghiệm. Điều này đòi hỏi mô hình không chỉ phải có khả năng tiếp thu và giữ thông tin từ ngữ cảnh mà còn cần khả năng suy luận và tìm kiếm thông tin chính xác để đưa ra câu trả lời đúng.

Nhiệm vụ này không chỉ giúp mô hình hiểu biết sâu sắc về nội dung văn bản mà còn đòi hỏi khả năng xử lý ngôn ngữ tự nhiên với độ chính xác cao. Ứng dụng của nhiệm vụ này rất đa dạng, từ việc cung cấp các hệ thống đánh giá trực tuyến trong lĩnh vực giáo dục đến việc tạo ra các công cụ hỗ trợ tìm kiếm thông tin hiệu quả trong các nguồn dữ liệu phong phú.

Qua nhiều năm phát triển, các mô hình đã được tiền huấn luyện như BERT và GPT đã chơi một vai trò quan trọng trong việc nâng cao khả năng thực hiện nhiệm vụ này, mang lại những đóng góp lớn cho sự tiến bộ của xử lý ngôn ngữ tự nhiên và ứng dụng của nó trong thế giới thực.

Ứng dụng Pre-trained model cho nhiệm vụ Multiple Choice để trả lời câu hỏi trắc nghiệm trên dữ liệu tiếng anh. Mục đích kế thừa khả năng hiểu được ngôn ngữ tiếng anh đã được đào tạo trước từ đó giảm thời gian training.

Thách thức của nhiệm vụ này;

- Để hiểu biết toàn diện: Mô hình cần biết rộng về ngữ cảnh để trả lời câu hỏi phức tạp chưa có trong bộ dữ liệu đào tạo.
- Sự đa nghĩa: Các câu hỏi và văn bản có nhiều ý nghĩa và đòi hỏi khả năng hiểu biết sâu sắc.

Nhiệm vụ trả lời câu hỏi trắc nghiệm là một trong những ứng dụng tiêu biểu của xử lý ngôn ngữ tự nhiên và đóng vai trò quan trọng trong việc tự động hóa quá trình đánh giá kiến thức và hiểu biết của mô hình về nội dung văn bản.

.2. Ứng dụng Pre-trained model cho nhiệm vụ trả lời câu hỏi trắc nghiệm

Nhiệm vụ Multiple Choice là một dạng phổ biến trong xử lý ngôn ngữ tự nhiên (NLP), trong đó mô hình cần đưa ra một lựa chọn chính xác từ giữa nhiều lựa chọn cho một câu hỏi cụ thể. Các pre-trained models trong NLP đã chứng minh rằng chúng có thể được fine-tuned hiệu quả cho nhiệm vụ này. Dưới đây là giới thiệu về cách áp dụng pre-trained models cho nhiệm vụ Multiple Choice:

Bước 1: Thử nghiệm và lựa chọn một pre-trained model phù hợp. BERT, RoBERTa, XLNet, ALBERT, và các mô hình khác đã được pre-trained trên dữ liệu lớn và có khả năng hiểu biểu diễn ngôn ngữ tự nhiên. Các model đã được training và cung cấp trên `huggingface`, `tensorflow`

Bước 2: Fine-tuning là quá trình điều chỉnh pre-trained model để nó có thể phù hợp với nhiệm vụ đề ra. Bạn cần một bộ dữ liệu huấn luyện chứa các cặp câu hỏi và lựa chọn, với nhãn cho câu trả lời đúng. Quá trình fine-tuning giúp mô hình hiểu cách phân loại lựa chọn trong ngữ cảnh của câu hỏi.

Bước 3: Chuẩn bị Dữ liệu là quan trọng để fine-tuning. Dữ liệu huấn luyện cần được chia thành tập huấn luyện và tập kiểm tra. Mỗi câu hỏi có thể có một hoặc nhiều lựa chọn, và mỗi lựa chọn đều cần được gắn nhãn để biết đó là câu trả lời đúng.

Bước 4: Huấn luyện và Đánh giá Thực hiện quá trình fine-tuning trên tập huấn luyện và sử dụng tập kiểm tra để đánh giá hiệu suất của mô hình. Thông thường, đánh giá đo lường độ chính xác (accuracy) hoặc các độ đo đánh giá khác tùy thuộc vào yêu cầu cụ thể của nhiệm vụ.

Bước 5: Sử dụng mô hình đã fine-tuned để dự đoán câu trả lời đúng cho các câu hỏi mới có lựa chọn.

Bước 6: Sử dụng thêm các kỹ thuật tăng cường thông tin RAG (Retrieval-Augmented Generation)

3.3. Các công cụ xây dựng

3.3.1. Sử dụng ngôn ngữ python

Python chơi một vai trò quan trọng trong lĩnh vực Trí tuệ nhân tạo (AI). Dưới đây là một số điểm mạnh và ứng dụng của Python trong lĩnh vực này:

Ngôn ngữ Lập trình Phổ biến: Python là ngôn ngữ lập trình phổ biến và được ưa chuộng trong cộng đồng AI. Điều này là do cú pháp đơn giản, dễ đọc, và tính linh hoạt của Python, giúp các nhà phát triển tập trung vào việc phát triển mô hình AI mà không mất nhiều thời gian với các chi tiết lập trình phức tạp.

Thư viện và Frameworks: Python có nhiều thư viện và frameworks mạnh mẽ hỗ trợ phát triển dự án AI. Các thư viện như NumPy và pandas hỗ trợ xử lý và phân tích dữ liệu, trong khi TensorFlow và PyTorch cung cấp các công cụ mạnh mẽ cho việc xây dựng và huấn luyện mô hình machine learning và deep learning.

Community Support: Cộng đồng Python rất đông đảo và tích cực, với nhiều nguồn tài nguyên và diễn đàn hỗ trợ. Điều này làm cho việc học và giải quyết vấn đề trong lĩnh vực trí tuệ nhân tạo trở nên thuận tiện.

Ứng dụng Rộng Rãi: Python được sử dụng trong nhiều ứng dụng của trí tuệ nhân tạo, bao gồm xử lý ngôn ngữ tự nhiên (NLP), thị giác máy tính, dự đoán và phân loại, và nhiều lĩnh vực khác.

Nhờ những ưu điểm này, Python đã trở thành ngôn ngữ lập trình hàng đầu cho nhiều dự án và nghiên cứu trong lĩnh vực trí tuệ nhân tạo.

3.3.2. Sử dụng công cụ Kaggle notebooks

Kaggle là một nền tảng trực tuyến nổi tiếng cho các dự án và cuộc thi trong lĩnh vực khoa học dữ liệu, machine learning và trí tuệ nhân tạo. Dưới đây là một số điểm quan trọng về Kaggle:

Cuộc Thi và Thách Thức (Competitions): Kaggle cung cấp nhiều cuộc thi và thách thức với dữ liệu thực tế từ các doanh nghiệp và tổ chức trên khắp thế giới. Người tham gia có cơ hội đặt câu hỏi, giải quyết các vấn đề thực tế và có cơ hội nhận các giải thưởng hấp dẫn.

Dữ Liệu: Kaggle cung cấp một kho dữ liệu lớn và đa dạng cho cộng đồng. Người dùng có thể tìm thấy các bộ dữ liệu từ nhiều lĩnh vực khác nhau như y học, tài chính, thị trường chứng khoán, thị giác máy tính, và nhiều lĩnh vực khác.

Notebooks và Kernels: Kaggle hỗ trợ sử dụng Jupyter Notebooks và Kernels, giúp người dùng chia sẻ, đọc và chạy mã nguồn một cách dễ dàng. Điều này thúc đẩy sự chia sẻ kiến thức và kinh nghiệm giữa cộng đồng.

Tài nguyên: Cung cấp môi trường training miễn phí với cấu hình cao cho việc training và thử nghiệm các thuật toán về trí tuệ nhân tạo.

- Hơn 100Gb bộ nhớ để lưu trữ dữ liệu.
- 30h sử dụng Gpu trong một tuần (GPU P100 15Gb , GPU T4x2 15Gb)
- CPU 30GB Ram không giới hạn thời gian.

Với sự linh hoạt và tính chất cạnh tranh, Kaggle đã trở thành một điểm đến quan trọng cho những người muốn thử thách bản thân và học hỏi trong lĩnh vực khoa học dữ liệu và trí tuệ nhân tạo.

3.3.3. Các thư viện sử dụng

Pandas là một thư viện Python cung cấp các cấu trúc dữ liệu và công cụ phân tích dữ liệu dễ sử dụng, đặc biệt là *DataFrame*. *DataFrame* là một bảng hai chiều với các hàng và cột có thể được gán nhãn, giúp quản lý và xử lý dữ liệu dạng bảng một cách thuận tiện.

NumPy là một thư viện Python cung cấp hỗ trợ cho các mảng và ma trận nhiều chiều. *NumPy* không chỉ giúp thực hiện các phép toán toán học và thống kê một cách hiệu quả trên dữ liệu số, mà còn là cơ sở cho nhiều thư viện khác, đặc biệt là trong lĩnh vực machine learning và scientific computing.

Datasets là một thư viện Python cung cấp các công cụ để quản lý và tải dữ liệu từ các nguồn khác nhau, đặc biệt là trong ngữ cảnh machine learning. Thư viện này cung cấp nhiều bộ dữ liệu chuẩn và dễ sử dụng để nghiên cứu và phát triển mô hình.

Transformers là một thư viện Python do Hugging Face phát triển, tập trung vào việc triển khai các mô hình xử lý ngôn ngữ tự nhiên (NLP) hiện đại, đặc biệt là các mô hình sử dụng kiến trúc transformer. Thư viện này cung cấp các mô hình đã được huấn luyện trước và các công cụ để tùy chỉnh mô hình cho nhiều nhiệm vụ NLP.

Sentence-transformers cung cấp các mô hình nhúng câu (sentence embeddings) hiệu quả, có thể được sử dụng trong nhiều ứng dụng như tìm kiếm semantec, phân loại văn bản, và clustering. Thư viện này dựa trên kiến trúc transformer để tạo ra các biểu diễn nhúng mạnh mẽ.

gc (Garbage Collector) là một mô-đun trong Python cung cấp giao diện đến trình quản lý bộ nhớ tự động của Python. Garbage collector là một cơ chế tự động thu dọn bộ nhớ, giúp giải phóng tài nguyên khi không còn cần thiết.

Faiss là một thư viện cung cấp các thuật toán hiệu quả cho việc tìm kiếm và phân loại trong không gian vector lớn. Đặc biệt, thư viện này thường được sử dụng trong các ứng dụng liên quan đến các vấn đề liên quan đến trí tuệ nhân tạo, đặc biệt là trong việc xử lý và tìm kiếm vector biểu diễn.

torch (*PyTorch*) là một thư viện máy học và deep learning phổ biến trong cộng đồng nghiên cứu và công nghiệp. Nó cung cấp một giao diện linh hoạt để xây dựng và huấn luyện mô hình, và được ưa chuộng trong việc triển khai các mô hình deep learning phức tạp. PyTorch cũng hỗ trợ tính toán trên GPU, giúp tăng tốc quá trình huấn luyện.

3.4. Giới thiệu về tập dữ liệu

Bộ dữ liệu Kaggle-LLM Science Exam (Use LLMs to answer difficult science questions). Bộ dữ liệu được sử dụng cho cuộc thi của kaggle <https://www.kaggle.com/competitions/kaggle-llm-science-exam/data>

Trong bộ dữ liệu công khai có 200 câu hỏi mẫu cùng với câu trả lời để hiển thị định dạng chúng về loại câu hỏi trong bộ kiểm tra. Mỗi câu hỏi bao gồm một prompt(câu hỏi), 5 lựa chọn gắn nhãn A, B, C, D, E và câu trả lời cũng được gắn nhãn answer.

Cuộc thi sử dụng bộ test ẩn ,dữ liệu chứa 4000 câu hỏi dùng để đánh giá kết quả của mô hình.

Các tập tin

train.csv - một bộ gồm 200 câu hỏi với cột câu trả lời

test.csv - bộ kiểm tra; Nhiệm vụ của bạn là dự đoán ba câu trả lời có khả năng xảy ra cao nhất khi đưa ra gợi ý.

sample_submission.csv - tệp gửi mẫu ở đúng định dạng

train.csv (181.26 kB)

Detail Compact Column

id	prompt	A	B	C	D	E	answer
0	Which of the following statements accurately describes the impact of Modified Newtonian Dynamics (MOND) on galaxy clusters?	MOND is a theory that reduces the observed missing baryonic mass in galaxy clusters by postulating t...	MOND is a theory that increases the discrepancy between the observed missing baryonic mass in galaxy...	MOND is a theory that explains the missing baryonic mass in galaxy clusters that was previously cons...	MOND is a theory that reduces the discrepancy between the observed missing baryonic mass in galaxy c...	MOND is a theory that eliminates the observed missing baryonic mass in galaxy clusters by imposing a...	D
1	Which of the following is an accurate definition of dynamic scaling in self-similar systems?	Dynamic scaling refers to the evolution of self-similar systems, where data obtained from snapshots ...	Dynamic scaling refers to the non-evolution of self-similar systems, where data obtained from snapsh...	Dynamic scaling refers to the evolution of self-similar systems, where data obtained from snapshots ...	Dynamic scaling refers to the non-evolution of self-similar systems, where data obtained from snapsh...	Dynamic scaling refers to the evolution of self-similar systems, where data obtained from snapshots ...	A
2	Which of the following statements accurately describes the origin and significance of the triskeles ...	The triskeles symbol was reconstructed as a feminine divine triad by the rulers of Syracuse, and lat...	The triskeles symbol is a representation of three interlinked spirals, which was adopted as an emble...	The triskeles symbol is a representation of a triple goddess, reconstructed by the rulers of Syracuse...	The triskeles symbol represents three interlocked spiral arms, which became an emblem for the rulers...	The triskeles symbol is a representation of the Greek goddess Hecate, reconstructed by the rulers of...	A
3	What is the significance of regularization ...	Regularizing the mass-energy of an electron	Regularizing the mass-energy of an electron	Regularizing the mass-energy of an electron	Regularizing the mass-energy of an electron	Regularizing the mass-energy of an electron	C

Hình 14 Bộ dữ liệu cuộc thi

Mục tiêu của cuộc thi

Lấy cảm hứng từ bộ dữ liệu OpenBookQA, cuộc thi này thách thức người tham gia trả lời các câu hỏi khó dựa trên cơ sở khoa học *được viết bằng Mô hình ngôn ngữ lớn*.

Công việc của bạn sẽ giúp các nhà nghiên cứu hiểu rõ hơn về khả năng tự kiểm tra của LLM và tiềm năng của LLM có thể chạy trong môi trường hạn chế về tài nguyên.

Bối cảnh

Khi phạm vi khả năng của mô hình ngôn ngữ lớn mở rộng, lĩnh vực nghiên cứu ngày càng phát triển là sử dụng LLM để mô tả đặc điểm của chúng. Bởi vì nhiều điểm chuẩn NLP hiện có đã được chứng minh là không quan trọng đối với các mô hình hiện đại, nên cũng có công trình thú vị cho thấy rằng LLM có thể được sử dụng để tạo ra nhiều nhiệm vụ thử thách hơn để kiểm tra từ trước đến nay những mô hình mạnh mẽ hơn.

Đồng thời, các phương pháp như lượng tử hóa và chất lọc kiến thức đang được sử dụng để thu nhỏ các mô hình ngôn ngữ một cách hiệu quả và chạy chúng trên phần cứng khiêm tốn hơn. Môi trường Kaggle cung cấp một góc nhìn độc đáo để nghiên cứu vấn đề này vì các bài gửi phải tuân theo cả giới hạn về GPU và thời gian.

Tập dữ liệu cho thử thách này được tạo bằng cách cung cấp các đoạn văn bản gpt3.5 về một loạt chủ đề khoa học được lấy từ wikipedia và yêu cầu nó viết một câu hỏi trắc nghiệm (với một câu trả lời đã biết), sau đó lọc ra các câu hỏi dễ.

Hiện tại, chúng tôi ước tính rằng các mô hình lớn nhất chạy trên Kaggle có khoảng 10 tỷ tham số, trong khi gpt3.5 có khoảng 175 tỷ tham số. Nếu một mô hình trả lời câu hỏi có thể vượt qua bài kiểm tra được viết bởi một mô hình viết câu hỏi có quy mô lớn hơn 10 lần thì đây sẽ là một kết quả thực sự thú vị; mặt khác, nếu một mô hình lớn hơn có thể đánh bại một mô hình nhỏ hơn một cách hiệu quả, thì điều này có ý nghĩa thuyết phục đối với khả năng LLM tự đánh giá và kiểm tra bản thân.

Bộ dữ liệu bổ sung cho cuộc thi

Bộ dữ liệu 60k-data-with-context-v2 bộ dữ liệu được tổng hợp từ các nguồn công khai từ kho lưu trữ Kaggle. <https://www.kaggle.com/datasets/cdeotte/60k-data-with-context-v2>

- 6.5k train examples for LLM Science Exam
- 15k high-quality train examples
- Science exam - use sci or not sci questions?
- Wikipedia STEM 1k
- llm-science-3k-data

- EduQG Dataset - LLM Science Exam Format (~3.4K)
- LLM Science Exam SciQ Dataset

Các tập dữ liệu được sinh bởi, gpt-3.5-turbo, tổng hợp ngẫu nhiên theo Wikipedia. Dữ liệu gồm một (prompt) câu hỏi, 5 lựa chọn gán nhãn A,B,C,D,E và câu trả lời được gán nhãn answer. Sau khi tổng hợp dữ liệu được gán thêm một ngữ cảnh của câu hỏi(Context). Context được trích xuất từ bộ dữ liệu wikipedia nhằm cung cấp thêm thông tin cho câu hỏi để tăng độ chính xác của mô hình và cung cấp thông tin về câu hỏi trong trường hợp câu hỏi thuộc nội dung mà mô hình chưa được học.

all_12_with_context2.csv (329.7 MB)

Detail Compact Column

prompt	context	A	B	C	D	E	answer	# source
In relation to Eunice Fay McKenzie's career, which statement accurately reflects her most notable wo...	Eunice Fay McKenzie (February 19, 1918 – April 16, 2019) was an American actress and singer. She als...	McKenzie showcased her singing talents in numerous productions, garnering critical acclaim.	McKenzie is primarily remembered for her starring roles opposite Gene Autry in popular Western films...	McKenzie gained recognition for her role as a child actress in a series of iconic silent films.	McKenzie's collaborations with director Blake Edwards were instrumental in her rise to fame.	McKenzie's successful career in sound films continued into adulthood, becoming known for her versati...	B	1
How does Modified Newtonian Dynamics (MOND) impact the observed "missing baryonic mass" discrepancy ...	The presence of a clustered thick disk-like component of dark matter in the Galaxy has been suggeste...	MOND is a theory that increases the discrepancy between the observed missing baryonic mass in galaxy...	MOND explains the missing baryonic mass in galaxy clusters that was previously considered dark matte...	MOND is a theory that reduces the observed missing baryonic mass in galaxy clusters by postulating t...	MOND is a theory that eliminates the observed missing baryonic mass in galaxy clusters by imposing a...	MOND's impact on the observed missing baryonic mass in galaxy clusters remains a subject of debate.	E	1
Which of the following statements accurately describes Ray Montgomerie's football career?	Woody Hartman is a retired American soccer goalkeeper and coach. Ray Willsey (September 30, 1928 – N...	Ray Montgomerie is a former footballer who played as a forward for Dundee United, Heart of Midlothia...	Ray Montgomerie is a former footballer who played as a defender for Dumbarton, Kilmarnock, and Parti...	Ray Montgomerie is a former footballer who played as a goalkeeper for Motherwell, Falkirk, and St. M...	Ray Montgomerie is a former footballer who played as a striker for Inverness Caledonian Thistle, Ros...	Ray Montgomerie is a former footballer who played as a midfielder for Celtic, Rangers, and Aberdeen.	B	1
What is the significance of the Museum of the Occupation of Latvia in Riga?	The Museum of the Occupation of Latvia () is a museum and historic educational institution located i...	The Museum of the Occupation of Latvia is a memorial dedicated to the victims of World War II in Lat...	The Museum of the Occupation of Latvia showcases the history of the Latvian independence movement du...	The Museum of the Occupation of Latvia was established in 1993 to exhibit artifacts and educate the ...	The Museum of the Occupation of Latvia primarily focuses on the cultural heritage of Latvia and its	The Museum of the Occupation of Latvia is a museum dedicated to the history of Riga and its developm...	C	1

Hình 15. Bộ dữ liệu được sinh bởi GPT có ngữ cảnh

Mô tả về các cột dữ liệu

prompt- văn bản của câu hỏi đang được hỏi

context - văn bản chứa ngữ cảnh để trả lời câu hỏi

A - tùy chọn A; nếu tùy chọn này đúng thì answer là A

B - tùy chọn B; nếu tùy chọn này đúng thì answer là B

C - tùy chọn C; nếu tùy chọn này đúng thì answer là C

D - tùy chọn D; nếu tùy chọn này đúng thì answer là D

E - tùy chọn E; nếu tùy chọn này đúng thì answer là E

answer - câu trả lời đúng nhất, được xác định bởi LLM tạo ra (một trong A, B, C, D hoặc E).

Ví dụ :

Prompt : Which of the following statements accurately describes the impact of Modified Newtonian Dynamics (MOND) on the observed "missing baryonic mass" discrepancy in galaxy clusters?

A: MOND is a theory that reduces the observed missing baryonic mass in galaxy clusters by postulating the existence of a new form of matter called "fuzzy dark matter."

B: MOND is a theory that increases the discrepancy between the observed missing baryonic mass in galaxy clusters and the measured velocity dispersions from a factor of around 10 to a factor of about 20.

C: MOND is a theory that explains the missing baryonic mass in galaxy clusters that was previously considered dark matter by demonstrating that the mass is in the form of neutrinos and axions.

D: MOND is a theory that reduces the discrepancy between the observed missing baryonic mass in galaxy clusters and the measured velocity dispersions from a factor of around 10 to a factor of about 2.

E: MOND is a theory that eliminates the observed missing baryonic mass in galaxy clusters by imposing a new mathematical formulation of gravity that does not require the existence of dark matter.

Answers: D

Dữ liệu Wikipedia

Wikipedia Plaintext (2023-07-01): Phiên bản mới nhất của bộ dữ liệu từ Wikipedia, cập nhật đến tháng 7 năm 2023, mang đến cho bạn một nguồn thông tin phong phú với 6.286.775 bài viết, tiêu đề, văn bản và danh mục. Ở đây mỗi bài viết được sắp xếp theo thứ tự chữ cái và số, và sau đó được đặt vào các tệp parquet tương ứng với ký tự đầu tiên của tiêu đề. Điều này dẫn đến việc có các tệp a-z, number (nếu tiêu đề bắt đầu bằng số), và other (nếu tiêu đề bắt đầu bằng ký hiệu). Điều này không chỉ giúp bạn tìm kiếm nhanh chóng mà còn tạo ra sự sắp xếp và tổ chức rõ ràng. Việc có thể truy cập thông tin theo cách này không chỉ là việc thuận tiện mà còn mang lại trải nghiệm tìm kiếm nhanh chóng và dễ dàng. Nó giúp thu thập thông tin từ Wikipedia theo yêu cầu cụ thể của mình. Với dung lượng tổng cộng lên đến 15.1GB việc tìm tổ chức theo các file nhỏ hơn cũng giảm thời gian xử lý thông tin từ đó tăng hiệu suất tính toán.

<https://www.kaggle.com/datasets/jjinho/wikipedia-20230701>

3.5. Xây dựng mô hình trả lời câu hỏi trắc nghiệm

Sử dụng bộ dữ liệu train.cv từ bộ dữ liệu *Kaggle-LLM Science Exam* làm dữ liệu test cho mô hình. Mục đích có 1 bộ dữ liệu đánh giá gần với dữ liệu test ẩn nhất từ đó cho kết quả đánh giá gần với kết quả của cuộc thi.

Sử dụng nguồn từ *60k-data-with-context-v2* được sinh ra từ gpt-3.5-turbo làm dữ liệu train cho mô hình.

Sử dụng bộ dữ liệu từ Wikipedia Plaintext (2023-07-01) để trích xuất ngữ cảnh cho mỗi câu hỏi.

3.5.1. Các phương pháp và mô hình đã thử nghiệm

a. Các bước xây dựng và thử nghiệm mô hình

Xây dựng mô hình và đánh giá khi chưa cho mô hình học ngữ cảnh của câu hỏi. Từ đó so sánh kết quả đạt được và lựa chọn mô hình. Các bước cơ bản xây dựng và thử nghiệm mô hình.

Bước 1: Chuẩn bị dữ liệu cho bài toán

Bước 2: Xác định hướng tiếp cận bài toán

Bước 3: Tiền xử lý dữ liệu

Bước 4: Xây dựng mô hình hoặc lựa chọn các mô hình pre-trained cho bài toán

Bước 5: Huấn luyện mô hình

Bước 6: Thử nghiệm trên bộ dữ liệu test

Bước 7: Submit kết quả cuộc thi và đánh giá kết quả.

b. Kết quả các mô hình đã thử nghiệm và lựa chọn mô hình.

- Thử nghiệm bài toán theo hướng multiclass classification. Tôi sử dụng mô hình pre-trained bert-base để encode từng câu hỏi và câu trả lời và đạt kết quả tốt nhất khi sử dụng thêm các kỹ thuật MultiHeadAttention, residual connections, Normalization layer và sử dụng 4 layer cuối trong đầu ra của bert-base cho mô hình. Sử dụng softmax là hàm kích hoạt cho đầu ra của mô hình để dự đoán xác suất trên mỗi đáp án. Mô hình với tổng 254.145.276 tham số, trong đó có 144.672.035 tham số được training và 109.482.241 tham số không cần huấn luyện lại. Kết quả tốt nhất khi submit cuộc thi 0.36.
- Thử nghiệm bài toán theo hướng binary classification. Tôi xem bài toán là đánh giá xem mỗi cặp câu hỏi và câu trả lời là đúng hay sai. Tôi sử dụng mô hình pre-trained bert-base để encode câu hỏi và câu trả lời và đạt kết quả tốt nhất khi sử dụng thêm các kỹ thuật MultiHeadAttention, residual connections, Normalization layer và sử dụng 4 layer cuối trong đầu ra của bert-base cho mô hình. Sử dụng sigmoid là hàm kích hoạt cho đầu ra của mô hình để dự đoán xác suất chính xác của cặp câu hỏi và câu trả lời này. Với mỗi câu hỏi lần lượt đưa mỗi cặp câu hỏi và câu trả lời qua mô hình và chọn ra đáp án có điểm xác suất chính xác cao nhất là đáp án đúng. Mô hình với tổng 138.416.642 tham số, trong đó có 28.934.401 tham số huấn luyện và 109.482.241 tham số không cần huấn luyện lại. Kết quả tốt nhất khi submit cuộc thi 0.45.
- Sử dụng nhiệm vụ multi-choice với mô hình pre-trained. Bài toán với 5 đầu vào mỗi đầu vào và tổng hợp của một cặp câu hỏi và câu trả lời. Model dựa vào dữ liệu training và kiến thức đã được training trước để đưa ra đáp án đúng. Tôi đã thử nghiệm trên một số mô hình và trọng số đã được training từ trước và public từ

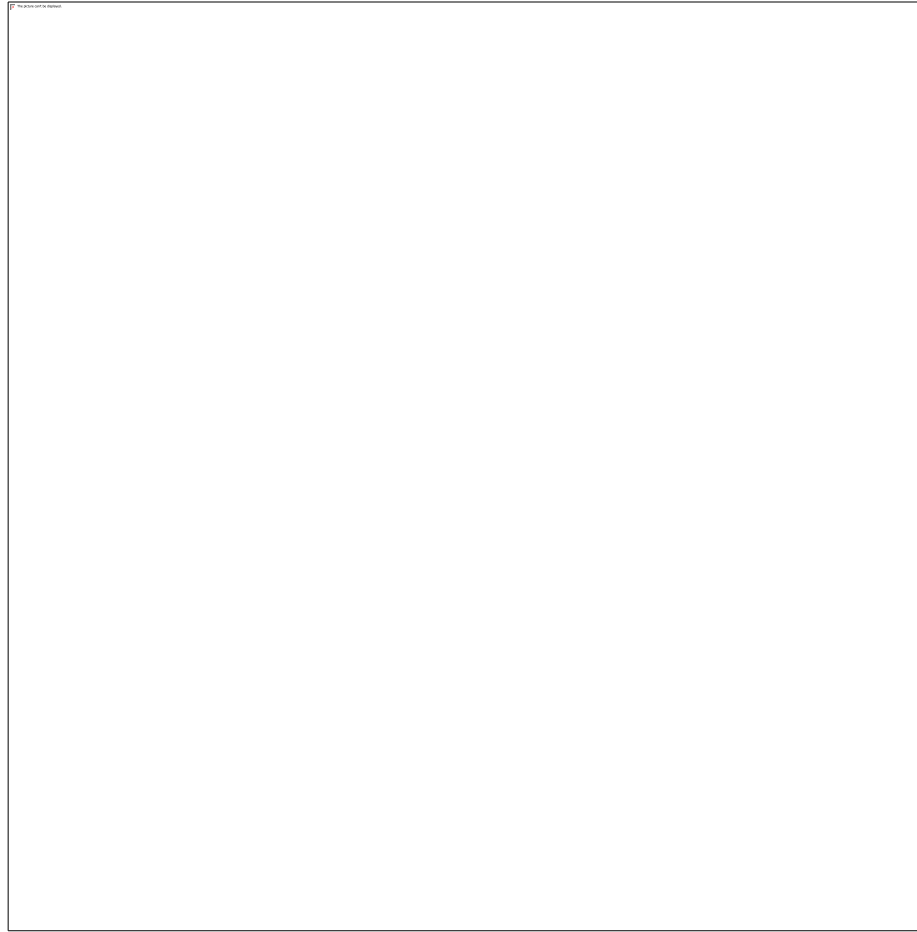
huggingface và tensorflow. Chúng tôi đã đạt được kết quả tốt nhất với model microsoft/deberta-v2-xlarge với 0.751 và 0.482 trên model electra của tensorflow.

- Tôi chọn tiếp cận bài toán theo nhiệm vụ multi-choice và sử dụng model microsoft/deberta-v2-xlarge làm model pre-trained chính cho bài toán. Tiếp theo huấn luyện lại mô hình với bộ dữ liệu 60k data có chứa ngữ cảnh của câu hỏi và lưu lại bộ trọng số đã huấn luyện.

3.5.2. Chiến lược cải thiện kết quả của mô hình

- Dữ liệu của cuộc thi đang thiếu ngữ cảnh của câu hỏi nên chúng tôi tìm dữ liệu ngữ cảnh cho câu hỏi nhằm tăng thông tin cho mô hình từ đó cải thiện kết quả của cuộc thi. Lựa chọn cải thiện RAG(Retrieval-Augmentend-Generation) với mục đích tìm kiếm thông tin của câu hỏi trong bộ dữ liệu wikipedia và trích xuất thông tin từ wikipedia từ đó sử dụng làm dữ liệu context bổ sung vào dữ liệu còn thiếu của cuộc thi.
- Chúng tôi xây dựng một bộ dữ liệu chứa vector biểu diễn của hơn 6 triệu bài viết trên wikipedia. Để giảm thời gian tính toán và tài nguyên chúng tôi chọn tiêu đề của bài viết và câu đầu tiên trong mỗi bài viết làm nội dung đặc trưng cho toàn bộ bài viết. Sử dụng model sentence-transformer để mã hóa vector đặc trưng và lưu thành file wikipedia_faiss.index để có thể dùng lại cho sau này. Sử dụng model sentence-transformer để mã hóa các câu hỏi kết hợp với các câu trả lời lấy ra vector đặc trưng cho từng từng câu hỏi từ đó sử dụng để tìm độ tương đồng của từng câu hỏi tới mỗi bài viết.
- Sử dụng thư viện faiss của facebookresearch
Faiss cung cấp việc tìm kiếm tương tự và phân cụm các vector bằng phương pháp như phân cụm KNN hay khoảng cách L2(Euclidean). Ở đây chúng tôi sử dụng việc phân cụm tìm ra các bài đăng có điểm số khoảng cách gần với câu hỏi nhất điểm số khoảng cách nhỏ nhất. Ở đây chúng tôi chọn lấy ra 5 bài viết có khoảng cách gần với câu hỏi nhất và lấy ra nội dung của 5 bài viết đó. Vì các bài viết có dung lượng và độ dài của bài viết còn lớn nên chúng tôi tiếp tục tìm kiếm thông tin của câu hỏi trong mỗi bài viết. Tách mỗi bài viết thành các câu đơn và đưa mỗi câu đó qua model sentence-transformer tạo vector biểu diễn và tiếp tục việc tính khoảng cách của mỗi câu hỏi tới các câu đơn trong 5 bài viết vừa chọn sau đó lấy ra 20 câu hỏi có khoảng cách câu hỏi nhất làm context. Từ đó thông tin đầu vào của mô hình đã có đủ ngữ cảnh cho phép model có đủ thông tin để suy luận kết quả. Đầu vào của model sẽ có context+prompt+ A (B or C or D or E).

3.5.3. Quy trình trả lời câu hỏi trắc nghiệm



Hình 16 Quá trình trả lời câu hỏi

Bước 1: Đưa câu hỏi và các đáp án trả lời vào mô hình

Bước 2: Tìm kiếm ngữ cảnh của câu hỏi và câu trả lời từ bộ dữ liệu wikipedia

Mã hóa câu hỏi và câu trả lời bằng model sentence-transformer. Từ đó đưa ra vector đặc trưng cho câu hỏi. Tìm kiếm các bài viết có có vector đặc trưng gần với vector đặc trưng của câu hỏi và câu trả lời và lấy ra 5 bài viết gần nhất. Trong 5 bài viết tách nhỏ thành các câu đơn và tìm kiếm 20 câu có độ tương đồng gần với câu hỏi và câu trả lời nhất làm ngữ cảnh của câu hỏi.

Bước 3: Xuất ra ngữ cảnh của câu hỏi

Ngữ cảnh của câu hỏi là tổng hợp của 20 câu có độ tương đồng cao nhất được tìm thấy.

Bước 4: Đưa câu hỏi kết hợp với ngữ cảnh truy xuất được vào mô hình dự đoán

Dữ liệu của câu hỏi bao gồm : câu hỏi , câu trả lời và ngữ cảnh được tìm được từ Wikipedia. Dữ liệu qua mô hình microsoft/deberta-v2-xlarge đã được training thêm với 60k data.

Bước 5: Đưa ra câu trả lời

Chọn câu đáp án có điểm số cao nhất làm kết quả của câu hỏi.

Chương 4: Kết quả và đánh giá

4.1. Xử lý dữ liệu

Dữ liệu với mỗi câu hỏi gồm: context, prompt,A,B,C,D,E,answer

Tiền xử lý dữ liệu với cấu trúc :

[CLS] + context + ##### + prompt + [STEP] + A + [STEP]

[CLS] + context + ##### + prompt + [STEP] + B + [STEP]

[CLS] + context + ##### + prompt + [STEP] + C + [STEP]

[CLS] + context + ##### + prompt + [STEP] + D + [STEP]

[CLS] + context + ##### + prompt + [STEP] + E + [STEP]

Mã hóa từ với tokenizer deberta với độ dài tối đa là 384

Mỗi lần tokenizer 60k data mất >30 phút để tiền xử lý vậy nên dữ liệu sau khi được tiền xử lý sẽ được lưu lại và tải lên kho lưu trữ của huggingface để lưu trữ và sử dụng để training cho những lần thử nghiệm sau.

https://huggingface.co/datasets/VuongQuoc/60k_dataset_multichoice_384?row=0

4.2. Xây dựng hàm đánh giá cho quá trình huấn luyện

Hàm đánh giá gồm : map_at_3 đánh giá theo top 3 đáp án đúng, đây cũng là cách đánh giá kết quả của mô hình theo thể lệ của cuộc thi. Và map_at_score đánh giá theo 1 kết quả đúng duy nhất.

```
def map_at_3(predictions, labels):
    map_sum = 0
    pred = np.argsort(-1*np.array(predictions),axis=1)[:,:3]
    for x,y in zip(pred,labels):
        z = [1/i if y==j else 0 for i,j in zip([1,2,3],x)]
        map_sum += np.sum(z)
```

```

return map_sum / len(predictions)

def map_at_score(predictions, labels):
    pred = np.argmax(predictions,axis = 1)
    bool_check = [(pred[i]==labels[i]) for i in range(len(pred))]
    count_true = np.sum(bool_check)
    return count_true/len(pred)

def compute_metrics(p):
    predictions = p.predictions.tolist()
    labels = p.label_ids.tolist()
    return {"map@3": map_at_3(predictions, labels),
            "accuracy":map_at_score(predictions, labels)}

```

4.3. Xây dựng model

Tính chỉnh các tham số training model pre-trained với model ban đầu được lựa chọn là *'microsoft/deberta-v3-large'*

```

training_args = TrainingArguments(
    warmup_ratio=0.1,
    learning_rate=2e-5,
    per_device_train_batch_size=1,
    per_device_eval_batch_size=2,
    num_train_epochs=1,
    report_to='none',
    output_dir = f'./checkpoints_{DATE}_microsoft_deberta_V{VER}',
    overwrite_output_dir=True,
    fp16=True,
    gradient_accumulation_steps=21,
    logging_steps=25,
    evaluation_strategy='steps',
    eval_steps=25,
    save_strategy="steps",
    save_steps=100,
    load_best_model_at_end=True,
    metric_for_best_model='accuracy',
    lr_scheduler_type='cosine',
    weight_decay=0.01,
    save_total_limit=2,
    push_to_hub = True,
    hub_token = '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',
)

```

Giải thích tham số:

warmup_ratio = 0.1 : Tôi lựa chọn tỉ lệ số bước được sử dụng cho giai đoạn warm-up so với tổng số bước đào tạo là 0.1.

learning_rate = 2e-5 : Tốc độ học ban đầu là 2e-5 (được lựa chọn qua việc training thử nghiệm với 1e-5, 2e-5, 3e-5, 2e-6 ...)

num_train_epochs = 1 : Số lượng epoch đào tạo là 1 (Lựa chọn qua việc training thử với 1, 2 epoch)

per_device_train_batch_size = 1, per_device_eval_batch_size = 2 : Kích thước batch cho mỗi lần train là 1 và eval là 2 (lựa chọn dựa trên giới hạn về phần cứng training)

gradient_accumulation_steps = 21 : Số bước cập nhật gradient trước khi thực hiện một bước cập nhật trọng số. (Lựa chọn qua việc thử nghiệm)

fp16: Để giảm dung lượng bộ nhớ và tăng tốc độ đào tạo.

logging_steps=25, evaluation_strategy='steps', eval_steps=25, save_strategy="steps", save_steps=100, save_total_limit=2 : Lựa chọn chiến lược đánh giá theo steps và cụ thể là 25 steps sẽ đánh giá kết quả 1 lần và cứ 100 steps sẽ lưu lại tham số tốt nhất. Giới hạn là 2 bộ tham số tốt nhất.

load_best_model_at_end=True : Tải lại bộ tham số tốt nhất khi kết thúc quá trình training.

weight_decay=0.01: Hệ số giảm trọng lượng (L2 regularization).

lr_scheduler_type='cosine' : Loại lịch trình tốc độ học là cosine.

push_to_hub = True : Tải model sau khi training lên kho lưu trữ huggingface để có thể sử dụng lại.

```
trainer = Trainer(
    model=model,
    args=training_args,
    tokenizer=tokenizer,
    data_collator=DataCollatorForMultipleChoice(tokenizer=tokenizer),
    train_dataset=dataset_load['train'],
    eval_dataset=dataset_load['test'],
    compute_metrics = compute_metrics,
    callbacks=[EarlyStoppingCallback(early_stopping_patience=5)],
)
```


```
trainer.train()
trainer.save_model(f'model_v{VER}')
```

Sử dụng bộ dữ liệu train với 60k data và test là 200 data. Sử dụng 200 data của cuộc thi làm dữ liệu test bởi vì tôi cho rằng nó là bộ dữ liệu gần với bộ dữ liệu ẩn nhất nên sẽ cho độ tin cậy tốt nhất cho cuộc thi.

Sử dụng `early_stopping_patience = 5` : Sau 5 lần đánh giá nếu model không cho kết quả tốt lên sẽ dừng training và lưu lại tham số tốt nhất.

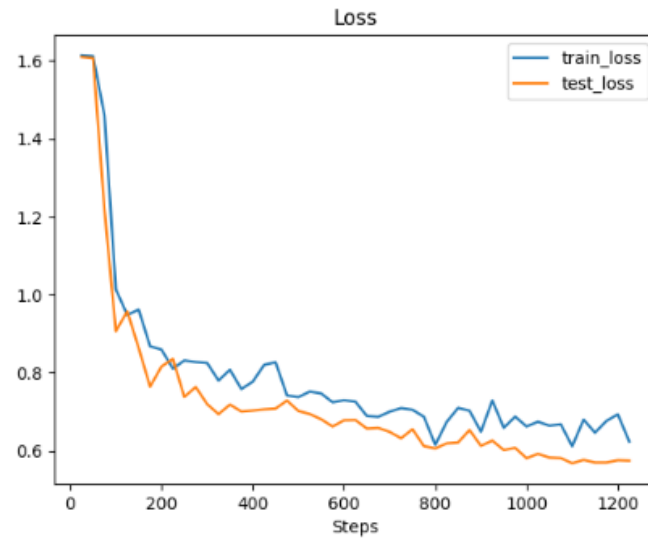
4.4.Kết quả đào tạo

Model đã được đào tạo trong 10h 24p và dừng tại 1225/1428 steps

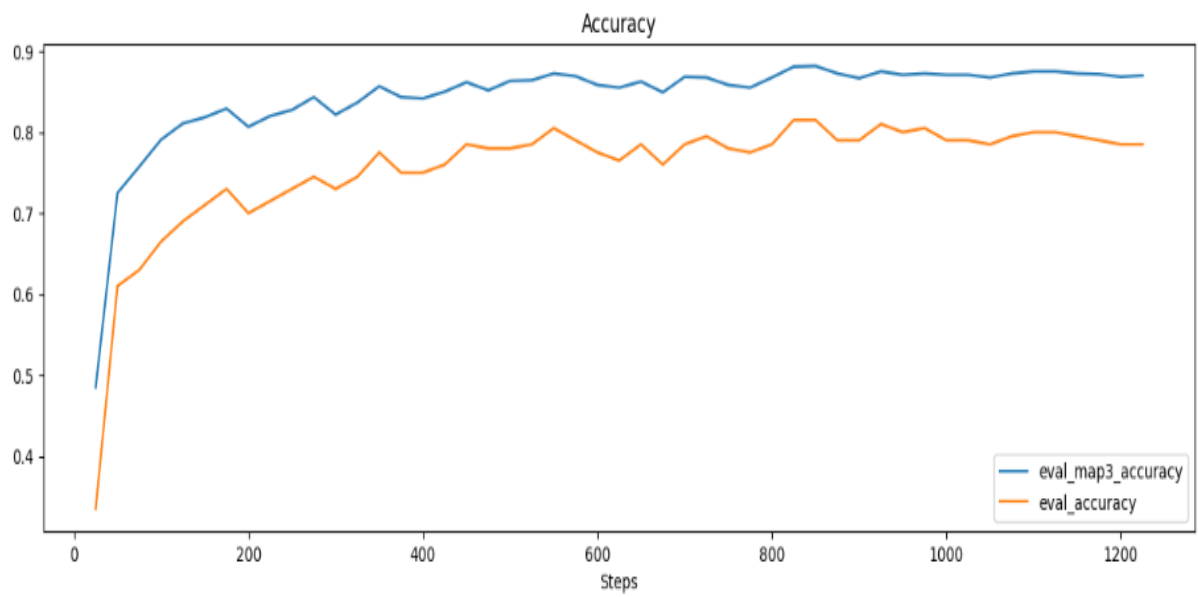


Step	Training Loss	Validation Loss	Map@3	Accuracy
25	1.613600	1.609221	0.485000	0.335000
50	1.611600	1.606294	0.725000	0.610000
75	1.459800	1.218634	0.757500	0.630000
100	1.013700	0.906754	0.790833	0.665000
125	0.948300	0.957444	0.810833	0.690000
150	0.961900	0.863439	0.818333	0.710000
175	0.867900	0.764401	0.829167	0.730000
200	0.859400	0.816143	0.806667	0.700000
225	0.810500	0.835504	0.820000	0.715000
250	0.831500	0.738066	0.827500	0.730000
275	0.827500	0.763591	0.843333	0.745000
300	0.825200	0.719643	0.821667	0.730000
325	0.780100	0.694006	0.836667	0.745000
350	0.807800	0.718544	0.856667	0.775000
375	0.758300	0.700706	0.843333	0.750000
400	0.777200	0.703194	0.841667	0.750000
425	0.820400	0.706209	0.850000	0.760000
450	0.826900	0.708248	0.861667	0.785000
475	0.741800	0.728756	0.851667	0.780000
500	0.737600	0.702062	0.863333	0.780000
525	0.751900	0.694266	0.864167	0.785000
550	0.746900	0.680736	0.872500	0.805000
575	0.724400	0.662249	0.869167	0.790000
600	0.729700	0.678261	0.858333	0.775000
625	0.725900	0.678812	0.855000	0.765000
650	0.689300	0.657071	0.862500	0.785000
675	0.687100	0.658699	0.849167	0.760000
700	0.700300	0.648503	0.868333	0.785000
725	0.709400	0.632038	0.867500	0.795000
750	0.705200	0.655397	0.858333	0.780000
775	0.687300	0.612138	0.855000	0.775000
800	0.615200	0.606050	0.867500	0.785000
825	0.674100	0.619088	0.880833	0.815000
850	0.709800	0.621344	0.881667	0.815000
875	0.702900	0.653335	0.872500	0.790000
900	0.648900	0.612704	0.866667	0.790000
925	0.728900	0.626148	0.875000	0.810000
950	0.658900	0.601896	0.870833	0.800000
975	0.687600	0.607593	0.872500	0.805000
1000	0.662400	0.581047	0.870833	0.790000
1025	0.674600	0.592205	0.870833	0.790000
1050	0.664400	0.582735	0.867500	0.785000
1075	0.668000	0.581355	0.872500	0.795000
1100	0.611500	0.568048	0.875000	0.800000
1125	0.679900	0.576690	0.875000	0.800000
1150	0.646600	0.569970	0.872500	0.795000
1175	0.676500	0.569964	0.871667	0.790000
1200	0.693600	0.575817	0.868333	0.785000
1225	0.623900	0.574785	0.870000	0.785000

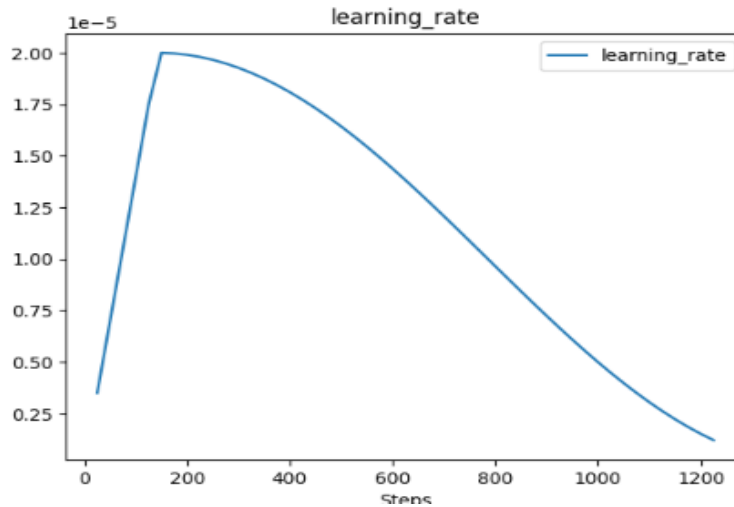
Hình 17 Hình ảnh kết quả đào tạo deberta



Hình 18 Biểu đồ $train_loss$ và val_loss



Hình 19 Biểu đồ đánh giá accuracy



Hình 20 Biểu đồ sự thay đổi của learning_rate

Qua nhiều lần training thử nghiệm tôi đã đạt được kết quả 81% accuracy và 88% cho top 3 lựa chọn đúng nhất trên tiêu chí đánh giá kết quả của cuộc thi. Sau khi sử dụng thêm kỹ thuật tăng cường cường thông tin(RAG) để trích xuất các thông tin có liên quan tới câu hỏi và câu trả lời kết quả đã cải thiện thêm 13% (không sử dụng RAG 75.1% cho top 3). Việc lựa chọn.

Sử dụng tăng cường (warm-up) trong tốc độ học là một kỹ thuật thường được áp dụng trong quá trình huấn luyện mô hình máy học, đặc biệt là trong các mô hình sử dụng tối ưu hóa gradient descent như Adam, SGD, hoặc các biến thể khác.

Tăng cường là quá trình tăng dần tốc độ học từ giá trị thấp lên giá trị mong muốn trong một số lượng bước huấn luyện ban đầu. Mục tiêu của kỹ thuật này là giảm khả năng mô hình "rơi vào" các cực tiểu cục bộ ở những giai đoạn đầu của quá trình học. Điều này có thể giúp mô hình hiệu quả hơn khi đối mặt với các vùng không gian lớn và phức tạp của không gian tham số.

Ưu điểm:
















Tránh cực tiểu cục bộ ở đầu quá trình học: Giúp mô hình tránh việc rơi vào cực tiểu cục bộ ở những giai đoạn đầu của quá trình học.

Stabilize quá trình học: Giúp ổn định quá trình học, đặc biệt là khi mô hình được khởi tạo với trọng số ngẫu nhiên và tốc độ học lớn.

Cải thiện khả năng tìm kiếm không gian tham số: Mở rộng khả năng tìm kiếm không gian tham số của mô hình, đặc biệt là khi có nhiều cực tiểu toàn cục.

4.5. Kết quả của quá trình tham gia cuộc thi

Đây là kết quả của quá trình bắt đầu và kết thúc cuộc thi mà chúng tôi đã đạt được (cuộc thi được diễn ra trong 3 tháng và mỗi ngày giới hạn nộp bài 3 lần)

Kaggle - LLM Science Exam				Late Submission	...
Overview Data Code Models Discussion Leaderboard Rules Team Submissions					
All Successful Selected Errors					
Recent					
Submission and Description		Private Score	Public Score	Selected	
 RAPIDS TF-IDF - [LB 0.904] - Version 2 Succeeded (after deadline) · QuocTuong · 3mo ago · Notebook RAPIDS TF-IDF - [LB 0.904] Version 2		0.890486	0.901581	<input type="checkbox"/>	
 DeBERTa Multichoice for Kaggle LLM Exam - Version 8 Succeeded · 08 Nguyễn Gia Bảo · 3mo ago · Notebook DeBERTa Multichoice for Kaggle LLM Exam Version 8		0.695217	0.715771	<input type="checkbox"/>	
 Sentence-transformer-Wiki - Version 9 Succeeded · QuocTuong · 4mo ago · Notebook Sentence-transformer-Wiki Version 9		0.785141	0.800249	<input checked="" type="checkbox"/>	
 Sentence-transformer-Wiki - Version 8 Succeeded · QuocTuong · 4mo ago · Notebook Sentence-transformer-Wiki Version 8		0.766124	0.786724	<input type="checkbox"/>	
 Sentence-transformer-Wiki - Version 7 Succeeded · QuocTuong · 4mo ago · Notebook Sentence-transformer-Wiki Version 7		0.768104	0.768622	<input type="checkbox"/>	
 Sentence-transformer-Wiki - Version 6 Succeeded · QuocTuong · 4mo ago · Notebook Sentence-transformer-Wiki Version 6		0.784984	0.787349	<input checked="" type="checkbox"/>	
 Sentence-transformer-Wiki - Version 5 for-kaggle-llm-exam?s... · 4mo ago · Notebook Sentence-transformer-Wiki Version 5		0.784984	0.787349	<input type="checkbox"/>	
 LLM-Competition - Version 3 Succeeded · Bao Nguyen Gia123 · 5mo ago · Notebook LLM-Competition Version 3		0.199124	0.213483	<input type="checkbox"/>	
 LLM_SCIENCE_EXAM - Version 17 Succeeded · QuocTuong · 5mo ago · Notebook LLM_SCIENCE_EXAM Version 17		0.202563	0.182272	<input type="checkbox"/>	
 LLM_SCIENCE_EXAM - Version 16 Succeeded · QuocTuong · 5mo ago · Notebook LLM_SCIENCE_EXAM Version 16		0.215692	0.217228	<input type="checkbox"/>	
 LLM_SCIENCE_EXAM - Version 15 Succeeded · QuocTuong · 5mo ago · Notebook LLM_SCIENCE_EXAM Version 15		0.21788	0.204744	<input type="checkbox"/>	
 LLM-Competition - Version 10 Succeeded · 08 Nguyễn Gia Bảo · 5mo ago · Notebook LLM-Competition Version 10		0.222882	0.22347	<input type="checkbox"/>	
 LLM_SCIENCE_EXAM - Version 14 Succeeded · QuocTuong · 5mo ago · Notebook LLM_SCIENCE_EXAM Version 14		0.21069	0.200998	<input type="checkbox"/>	
 LLM_SCIENCE_EXAM - Version 13 Succeeded · QuocTuong · 5mo ago · Notebook LLM_SCIENCE_EXAM Version 13		0.212566	0.208489	<input type="checkbox"/>	
 LLM_SCIENCE_EXAM - Version 12 Succeeded · QuocTuong · 5mo ago · Notebook LLM_SCIENCE_EXAM Version 12		0.21788	0.200998	<input type="checkbox"/>	

Hình 21. Kết quả quá trình nộp bài

Trong cuộc thi này điểm số của tôi đã cải thiện từ (0.2--> 0.8) với hơn 100 submissions. Thứ hạng đạt được trong cuộc thi 1224 / 2664.

Khi kết thúc cuộc thi tôi đã thử sử dụng model deberta của mình training và với code của của người chơi xếp hạng 10 của cuộc thi này với model deberta và đạt được độ chính xác top 3 là 0.92. Mô hình của tôi cũng đã đạt được 1 kết quả là 0.9 trong cuộc thi này. Tôi nhận thấy mô hình của mình không quá kém so với với những mô hình trên. Tuy nhiên người chơi này đã sử dụng thêm 1 nguồn dữ liệu từ *STEM wikipedia subset based on Cohere embeddings* để tăng cường thông tin cho câu hỏi. Sử dụng TF-IDF và một số kỹ thuật nhằm tăng cường độ chính xác cho đầu ra của quá trình trích xuất thông tin bổ sung.

Về top 1 của cuộc thi đã đạt được kết quả 0.93, đây là một team tốt nhất từ lúc bắt đầu cuộc thi. Theo như bài chia sẻ của họ về cuộc thi này: Ngay từ đầu việc thử nghiệm các mô hình nhỏ như deberta không phải mục tiêu chính. Họ đã xác định việc sử dụng nhiều đường dẫn tìm kiếm thông tin(RAG) , ở đây họ đã sử dụng 3 luồng thông tin với thời gian cho mỗi lần nộp là >2h tuy nhiên thời gian nộp của tôi chỉ 10p. Tiếp theo là việc họ đã xác định việc chạy mô hình LLM > 7b tham số (Llama-2-7b, mistral-7b, xgen-7b-8k-base, Llama-2-13b). Nó là điều mà tôi không thể nghĩ là nó có thể làm được trên bộ tài nguyên giới hạn của Kaggle(2 GPU 15gb) và các model lớn hơn 100GB , tài nguyên đầu vào đã sử dụng là 2.46TB. Ở đây họ đã tách các mô hình lớn này thành từng lớp nhỏ và đưa dữ liệu qua từng lớp nhỏ này thay vì tải toàn bộ model lên(đó là điều không thể). Đây là một kết quả tốt và xứng đáng của top 1. Tôi cũng học hỏi được thêm nhiều về cách giải quyết bài toán tại cuộc thi này.

Kết luận

1. Kinh nghiệm từ cuộc thi

- Xác định nhiệm vụ của mô hình: Đảm bảo rõ ràng về mục tiêu và nhiệm vụ chính mà mô hình sẽ thực hiện, ví dụ như phân loại, dự đoán, hay sinh văn bản.
- Lựa chọn mô hình cơ sở trước khi tối ưu hóa: Việc chọn một mô hình cơ sở trước khi tối ưu hóa với RAG giúp giảm thời gian thử nghiệm và đảm bảo mô hình có hiệu suất tốt ngay từ đầu.
- Tiền xử lý dữ liệu và sử dụng dataset: Sử dụng dataset với cơ chế memory mapping để tối ưu thời gian tiền xử lý dữ liệu thông qua xử lý song song, giảm bớt thời gian cần thiết.
- Lưu trữ dữ liệu sau tiền xử lý trên Hugging Face: Lưu trữ dữ liệu sau khi tiền xử lý trên Hugging Face giúp giảm thời gian trong mỗi lần huấn luyện, đặc biệt quan trọng với lượng lớn dữ liệu.
- Quản lý dữ liệu từ Wikipedia: Tối ưu hóa quá trình tìm kiếm context và giảm lượng tài nguyên bằng cách loại bỏ dữ liệu không cần thiết, giải phóng bộ nhớ, và chia nhỏ các file dữ liệu từ Wikipedia.
- Tìm kiếm thêm dữ liệu từ các cuộc thi và nguồn tin khác: Đọc và sử dụng dữ liệu từ các cuộc thi trước giúp mô hình học được từ nhiều nguồn và tăng cường dữ liệu huấn luyện.
- Sử dụng truy xuất dữ liệu tăng cường RAG: Học cách truy xuất dữ liệu tăng cường RAG để cải thiện khả năng đề xuất các câu trả lời chính xác.
- Sử dụng layer và embedding đóng băng để giảm tham số huấn luyện: Tối ưu hóa mô hình bằng cách sử dụng layer và embedding đóng băng để giảm số lượng tham số và thời gian huấn luyện.
- Nghiên cứu các tài liệu và kết quả liên quan: Tìm kiếm và đọc các tài liệu liên quan từ cuộc thi và thảo luận để cập nhật kiến thức, lấy kinh nghiệm từ các phương pháp hay kết quả đã được thử nghiệm trước đó. Kiểm tra và tham khảo các kết quả xuất sắc: Nên xem xét các kết quả tốt nhất từ các cuộc thảo luận trước để hiểu rõ về khả năng đạt được và điều chỉnh mô hình để cải thiện hiệu suất.

2. Phương hướng phát triển

- Tối ưu hóa mô hình và thử nghiệm thêm các mô hình mới để cải thiện hiệu suất của mô hình
- Mở rộng huấn luyện và có thể ứng dụng cho ngôn ngữ tiếng việt

- Thử nghiệm thêm các kỹ thuật tăng cường dữ liệu khác giúp giảm thời gian tìm kiếm ngữ cảnh và có độ chính xác cao
- Từ những kinh nghiệm đã đạt được sẽ tham gia thêm các thử thách khác.

Tài liệu tham khảo

<https://product.vinbigdata.org/cac-ky-thuat-tach-tu-trong-xu-ly-ngon-ngu-tu-nhien/>

<https://machinelearningmastery.com/the-transformer-attention-mechanism/>

<https://www.kaggle.com/code/quctngngvng/train-model-with-context-27-9>

<https://www.kaggle.com/code/quctngngvng/llm-science-exam-bert-base-attention?scriptVersionId=139842231>

<https://discuss.huggingface.co/t/how-to-get-the-loss-history-when-use-trainer-train/17486/4>

<https://medium.com/@tavishi.1402/what-is-perception-in-neural-network-fd22a84c08ea>

<https://www.kaggle.com/datasets/jjinho/wikipedia-20230701>

<https://github.com/facebookresearch/faiss>

<https://www.geeksforgeeks.org/how-to-calculate-cosine-similarity-in-python/>