TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC

PHÁT HIỆN CÂU HỎI TƯƠNG ĐỒNG TRONG LĨNH VỰC Y TẾ

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC

Chuyên ngành: Toán Tin

Chuyên sâu: Tin học

Giảng viên hướng dẫn: **TS. LÊ HẢI HÀ**

Chữ ký của GVHD

Sinh viên thực hiện: **NGUYỄN ĐỨC THẮNG**

Lớp: **KSTN Toán Tin - K61**

HÀ NỘI, 06/2021

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

1. Mục đích và nội dung của đồ án:

2. Kết quả đạt được:

3. Ý thức làm việc của sinh viên:

Hà Nội, ngày tháng 06 năm 2021

Giảng viên hướng dẫn
(Ký và ghi rõ họ tên)

Lời mở đầu

Trong những năm gần đây, trí tuệ nhân tạo ngày càng phát triển và đi vào cuộc sống hàng ngày. Trong đó, xử lý ngôn ngữ tự nhiện (NLP) là một nhánh của trí tuệ nhân tạo tập trung vào các ứng dụng trên văn bản và ngôn ngữ, âm thanh. Trong trí tuệ nhân tạo thì xử lý ngôn ngữ tự nhiên là một trong những phần khó, vì để giúp máy hiểu được ngôn ngữ không phải là đơn giản. Các bài toán của xử lý ngôn ngữ bao gồm: chuyển văn bản sang giọng nói và ngược lại, phát hiện lỗi chính tả, tóm tắt văn bản, phân tích cảm xúc...

Hàng loạt các ứng dụng của xử lý ngôn ngữ tự nhiên đang ngày càng phát triển ra đời. Trong đó, lĩnh vực y tế là lĩnh vực khó, đòi hỏi có kiến thức chuyên môn. Sự phát triển của internet cũng thay đổi xu hướng của người dùng. Mọi người ngày càng có xu hướng tìm kiếm câu hỏi của mình trên mạng. Tuy nhiên, số lượng người đặt câu hỏi thường rất nhiều so với số lượng người giải đáp. Các câu hỏi thường xuyên bị lặp lại về ngữ nghĩa. Vì vậy, việc xây dựng một hệ thống tự động trả lời các câu hỏi về lĩnh vực y tế là điều cần thiết, giúp người dùng có được giải đáp nhanh chóng nếu câu hỏi đã tồn tại, và đồng thời giúp các chuyên gia, bác sĩ có được đề xuất ưu tiên với các câu hỏi chưa xuất hiện.

Trong đồ án này, em nghiên cứu sử dụng các thuật toán, mô hình trong trí tuệ nhân tạo để giải quyết bài toán trên. Nội dung đồ án được chia làm 3 chương:

- **Chương 1:** Tổng quan về bài toán, mục tiêu và quá trình thực nghiệm đồ án.
- **Chương 2:** Trình bày về các kiến thức cơ sở, lý thuyết về các thuật toán và mô hình sẽ sử dụng để thực nghiệm cho bài toán.
- **Chương 3:** Mô tả chi tiết quá trình thực nghiệm, xây dựng và phân tích dữ liệu, kết qủa thực nghiệm các mô hình.

Mặc dù đã cố gắng rất nhiều, nhưng với kiến thức và thời gian hạn chế nên không thể tránh khỏi những thiếu sót. Rất mong được sự góp ý của thầy cô và bạn bè để đồ án được hoàn thiện hơn.

Lời cảm ơn

Đi qua những năm tháng Bách Khoa, ta mới biết tuổi trẻ đáng trân trọng như thế nào. Trân trọng, là vì có những lúc khó khăn tưởng chừng như gục ngã, nhưng sau tất cả, ta lại thấy mình trưởng thành biết bao nhiêu. Năm năm học ở Bách Khoa Hà Nội, đó không phải là khoảng thời gian quá dài trong cuộc đời của mỗi con người. Nhưng những năm tháng này là những năm tháng ý nghĩa không bao giờ quên với em. Em học được nhiều điều từ ngôi trường này. Tại đây, em được dẫn dắt bởi những người thầy cô tràn đầy tâm huyết với sinh viên. Từ lãnh đạo nhà trường cho đến những viện, phòng ban đều dành cho sinh viên những sự chăm sóc tỉ mỉ nhất. Tại đây, em được quen những người bạn giỏi để cùng nhau học tập, cùng nhau vui chơi hết mình và xây dựng những kỉ niệm đẹp. Và cũng tại ngôi trường này, em được rèn luyện cả về kỹ năng cứng và kỹ năng mềm để có thể trở thành một kỹ sư với một nền tảng chắc chắn cho tương lai.

Em xin cảm ơn lãnh đạo nhà trường cùng các thầy cô đã giúp đỡ em trong suốt thời gian qua. Em cũng xin cảm ơn thầy Lê Hải Hà, giảng viên đại học Bách Khoa Hà Nội - người tận tình hướng dẫn em suốt từ đồ án I cho đến đồ án tốt nghiệp này. Thầy đã ân cần chỉ dạy cho em rất nhiều kiến thức quý báu. Em xin gửi lời cảm ơn đến Viện Toán ứng dụng và Tin học, các thầy cô cùng toàn thể các bạn lớp kỹ sư tài năng Toán Tin khóa K61 đã sát cánh, cùng em vượt qua những khó khăn trong quá trình học tập.

Diều may mắn của em là trong quá trình làm đồ án tốt nghiệp đã có được môi trường làm việc lý tưởng cho việc học hỏi và nghiên cứu - Công ty DeepCare Việt Nam. Em xin bày tỏ sự biết ơn đến anh Nguyễn Văn Hạnh - giám đốc công ty Deepcare, thầy Nguyễn Kiêm Hiếu - giảng viên đại học Bách Khoa Hà Nội, cùng các thành viên khác của công ty đã hỗ trợ em trong suốt thời gian làm việc nghiên cứu tại đây.

Em xin cảm ơn tới công ty FPT Software, trung tâm nghiên cứu FSoft AI Lab, các anh chị và các bạn trong chương trình AI Residency đã chỉ dạy em nhiều kiến thức, tạo điều kiện thuận lợi cho em trong quá trình nghiên cứu và làm đồ án.

Cuối cùng, con xin cảm ơn bố mẹ, cảm ơn gia đình, những người thân yêu đã là điểm tựa vững chắc cho con trong cuộc sống này. Sự hy sinh âm thầm, cao cả, tình yêu thương từ bố mẹ và gia đình đã là nguồn động viên cho con trưởng thành hôm nay và mai sau.

Danh mục kí hiệu và các thuật ngữ

Từ viết tắt	Từ chuẩn	Diễn giải
CNN	Convolution Neural Network	Mạng Neural tích chập
EFB	Exclusive Feature Bundling	
FC	Fully Connected	Lớp kết nối dày đặc
GBDT	Gradient Boosting Decision Tree	
GOSS	Gradient-based One-Side Sampling	
MLM	Masked Language Model	Mô hình ngôn ngữ đánh dấu
NLP	Natural Language Process	Xử lý ngôn ngữ tự nhiên
NSP	Next Sentence Prediction	Dự đoán câu kế tiếp

Mục lục

Danh sách hình vẽ

Chương 1

Tổng quan về bài toán

1.1 Lý do chọn đề tài

Với sự xuất hiện của internet và sự xuất hiện của các trang web trả lời câu hỏi. Mọi người ngày càng tìm kiếm online nhiều các câu trả lời cho câu hỏi của họ. Tuy nhiên, các lĩnh vực yêu cầu tính chuyên môn cao như y tế thường có số lượng người đặt câu hỏi vượt xa số lượng câu trả lời. Và trong tiếng Việt thì vẫn chưa phổ biến. Các câu trả lời cho lĩnh vực y tế đòi hỏi các chuyên gia hoặc các bác sĩ có chuyên môn để giải đáp được, các câu hỏi cũng thường xuyên lặp lại về mặt ngữ nghĩa. Một cách để khắc phục vấn đề này là xây dựng hệ thống trả lời tự động so sánh các câu hỏi chưa được trả lời với các câu hỏi đã được trả lời có cùng ý nghĩa để từ đó đưa ra câu trả lời thích hợp hoặc đánh dấu chúng là ưu tiên cho một bác sĩ nếu không có câu trả lời tồn tại cho câu hỏi đó. Phương pháp này giúp các bác sĩ tiết kiệm thời gian hơn, giảm thiểu số lượng câu hỏi chưa được trả lời và giảm chi phí chăm sóc trực tuyến, đồng thời cũng đáp ứng tốt nhu cầu của người đặt câu hỏi.

Tuy nhiên, nghiên cứu một thuật toán chính xác để tìm các câu hỏi tương đồng là bất khả thi. Các thuật toán sử dụng Word embedding sau đó đo độ tương đồng thường không hiệu quả, đặc biệt là cho tiếng Việt. Đây gọi là các phương pháp học không giám sát, các phương pháp này thường chưa học được ngữ nghĩa sâu bên trong của ngôn ngữ.

Với sự phát triển của xử lý ngôn ngữ tự nhiên. Ngày càng có nhiều ứng dụng thông minh phục vụ cho cuộc sống của con người trong lĩnh vực ngôn ngữ, âm thanh. Tuy nhiên, việc áp dụng xử lý ngôn ngữ tự nhiên cho những lĩnh vực chuyên sâu như y tế vẫn chưa phổ biến. Một khó khăn cho các nghiên cứu mảng này là không có bộ dữ liệu tốt để thực nghiệm. Với những bất cập này, đồ án này tập trung nghiên cứu bài toán xây dựng hệ thống hỏi đáp trong lĩnh vực y tế dựa trên câu hỏi tương đồng. Đồ án tập trung xây dựng bộ dữ liệu có nhãn cho tiếng Việt và thực nghiệm so sánh chất lượng của mô hình. Cuối cùng, lựa chọn ra mô hình thích hợp và thiết kế thành hệ thống ứng dụng thực tiễn.

1.2 Mục tiêu của đề tài

Đồ án hướng tới xây dựng bộ dữ liệu tiếng Việt và mô hình nhằm giải quyết bài toán đặt ra ở phần trên. Hệ thống này phải có khả năng xử lý dữ liệu lớn và liên tục trong một khoảng thời gian nhỏ vừa đủ, đảm bảo yêu cầu thời gian thực để tích hợp vào hệ thống chatbot.

Bộ dữ liệu được xây dựng phải đảm bảo đủ lớn và tốt để cho mô hình có thể học. Độ chính xác của mô hình phải cao và tốc độ xử lý phải nhanh. Ngoài ra, đồ án cũng thiết kế một khung chuẩn chung để có thể dễ dàng cải tiến mô hình và bổ sung thêm dữ liệu sau này.

Chi tiết các bước thực hiện đề tài như sau:

1. Thu thập dữ liệu
   - Phân tích, tìm hiểu cấu trúc dữ liệu cần xây dựng.
   - Thu thập dữ liệu từ các trang hỏi đáp về lĩnh vực y tế trên internet.
2. Gán nhãn dữ liệu
   - Xây dựng hệ thống gãn nhãn dữ liệu.
   - Chỉnh sửa lại nội dung dữ liệu đã thu thập nếu có sai sót.
3. Thực nghiệm mô hình
   - Phân tích dữ liệu.
   - Tìm hiểu các thuật toán và đánh giá thử nghiệm. So sánh hiệu năng về tốc độ và độ chính xác.
   - Lựa chọn thuật toán phù hợp.

Chương 2

Cơ sở lý thuyết

2.1 Mô hình CNN

Convolutional Neural Network (CNN – Mạng neural tích chập) là một trong những mô hình học sâu tiên tiến. Nó giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao. CNN được sử dụng nhiều trong các bài toán nhận dạng các đối tượng trong ảnh. Ngoài ra, CNN cũng được sử dụng trong xử lý ngôn ngữ tự nhiên.

2.1.1 Cấu trúc mạng CNN

Mạng CNN là một tập các lớp tích chập (convolution) chồng lên nhau và sử dụng các hàm phi tuyến như ReLU, tanh, sigmoid, ... là hàm kích hoạt. Càng những tầng sau của mạng CNN thì càng trích lọc những thông tin chi tiết và sâu hơn trong ảnh. Trong mô hình CNN, các lớp của mạng liến kết với nhau thông qua cơ chế tích chập. Lớp tiếp theo là kết quả tích chập từ lớp trước đó, nhờ vậy mà ta có được các kết nối cục bộ của các phần của ảnh với nhau.

Mạng CNN có kiến trúc khác với mạng neural thông thường. Mạng neural thông thường chuyển đầu vào thông qua hàng loạt tầng ẩn. Mỗi tầng là tập các neural và các tầng liên kết đầy đủ với nhau. Và tầng cuối cùng là tầng sẽ trả ra kết quả đại diện cho dự đoán của mạng. Tuy nhiên, mạng neural thông thường không đáp ứng được trong việc trích lọc đặc trưng ảnh vì nếu chỉ một ảnh đầu vào kích thước 3000×3000 pixel thì riêng tầng đầu vào của mạng neural đã cần 9000000 units. Nếu cứ xây dựng mạng neural cho xử lý ảnh sẽ tạo ra một khối lượng tính toán rất lớn. Dựa vào tính chất của ảnh, những pixel ở một cụm gần nhau có thể mang thông tin ý nghĩa như nhau, CNN ra đời giúp lấy thông tin cục bộ trong một vùng ảnh (subsampling) của ảnh nhưng khối lượng tính toán ít hơn. Về cơ bản, mạng CNN gồm 2 thành phần:

- **Phần trích xuất đặc trưng:** Trong phần này, mạng sẽ tiến hành tính toán hàng loạt các phép tích chập (convolution) và phép hợp nhất (pooling) để phát hiện đặc trưng. Ví dụ: nếu ta có ảnh khẩu súng thì phần này sẽ nhận dạng đặc

điểm chung và chi tiết của một khẩu súng như ngòi súng, màu súng, ...

- **Phần phân lớp:** Tại phần này, một lớp với các liên kết đầy đủ sẽ đóng vai trò như một bộ phân lớp các đặc trưng đã trích xuất được trước đó. Tầng này sẽ đưa ra xác suất của một đối tượng trong hình. Số units tầng cuối của tầng phân lớp phải bằng số lớp mà chúng ta cần phân loại.

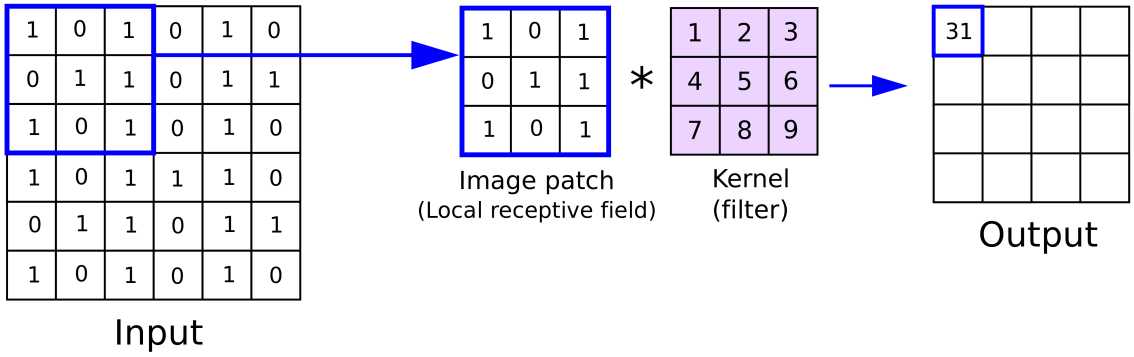
2.1.2 Lớp tích chập

Lớp tích chập (convolution layer) thường là lớp đầu tiên trong mô hình CNN. Lớp này có chức năng phát hiện ra các đặc trưng về không gian một cách hiệu quả.

Filter

Filter (hay còn gọi là kernel) là một trong những tham số quan trọng trong CNN. Kích thước filter trong tầng tích chập phổ biến hiện nay là 3×3. Kích thước filter thường là số lẻ, ví dụ 3×3, 5×5, 7×7.

Kích thước filter thường không quá lớn. Vì kích thước nhỏ nó có thể trích xuất cục bộ chi tiết hơn, kích thước ảnh giảm chậm hơn. Ta thực hiện các phép tích chập bằng cách trượt filter từ trái sang phải, từ trên xuống dưới theo dữ liệu đầu vào. Tại mỗi vị trí, ta tiến hành phép nhân ma trận và tính tổng tất cả các giá trị để đưa vào bản đồ đặc trưng.

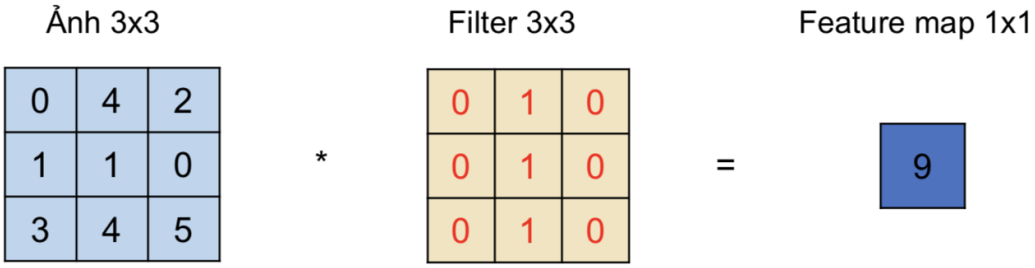


Hình 2.1: Ví dụ về phép tích chập với kernel/filter 3×3 trên ảnh

Với ảnh RGB có 3 kênh màu red, green, blue thì filter phải có cùng độ sâu (depth) với ảnh.

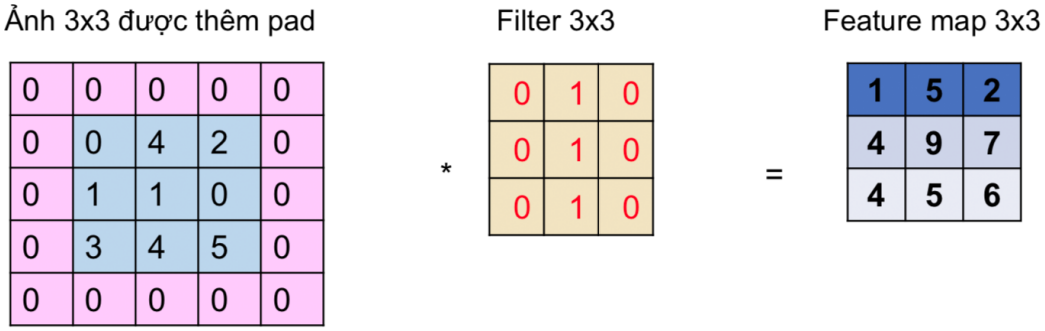
2.1.3 Padding

Khi dùng các phép tích chập, thông tin ở biên bức ảnh bị biến mất và kích thước của ảnh giảm nhanh chóng (Hình 2.2).



Hình 2.2: Kích thước ảnh giảm nhanh chóng khi dùng tích chập

Để khắc phục vấn đề này, chúng ta sử dụng padding. Bằng việc thêm các giá trị 0 vào biên, ta sẽ có *zero padding* (Hình 2.3).



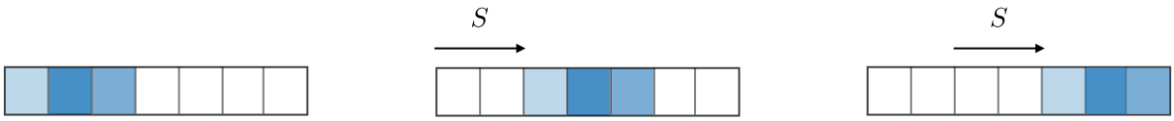
Hình 2.3: Thêm padding vào ảnh

Sau khi thêm padding chúng ta có một số lợi thế như sau:

- Không mất mát thông tin viền nên nhận diện sẽ tốt hơn, tìm được chính xác đối tượng hơn.
- Đầu ra của CNN kích thước sẽ giảm dần nên khi thêm padding sẽ giúp giảm chậm hơn.

Stride

Đối với phép convolution hay pooling thì stride (S) là độ dài bước trượt của filter. Như trong hình 2.4, độ dài bước trượt $S = 2$.

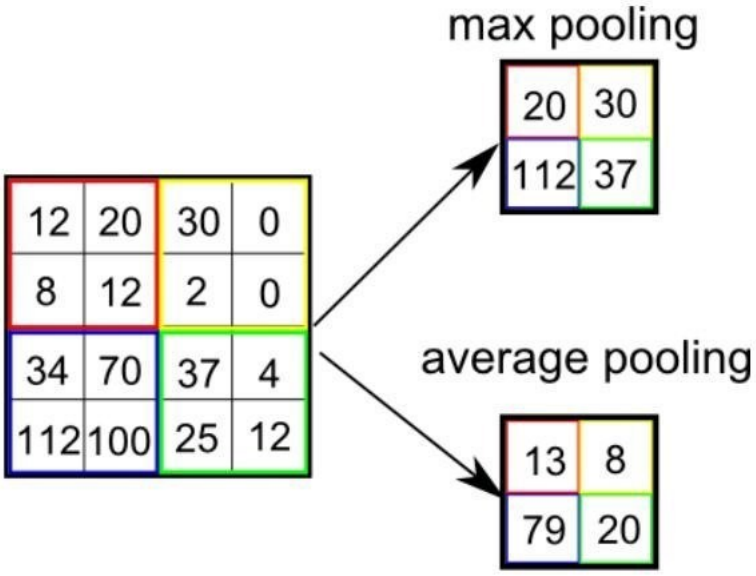


Hình 2.4: Minh hoạ cho stride

2.1.4 Lớp pooling

Lớp pooling thường được dùng giữa các lớp tích chập, để giảm kích thước dữ liệu nhưng vẫn giữ được các thuộc tính quan trọng. Kích thước dữ liệu giảm giúp giảm việc tính toán trong mô hình.
Có 2 loại pooling thường được sử dụng trong CNN:

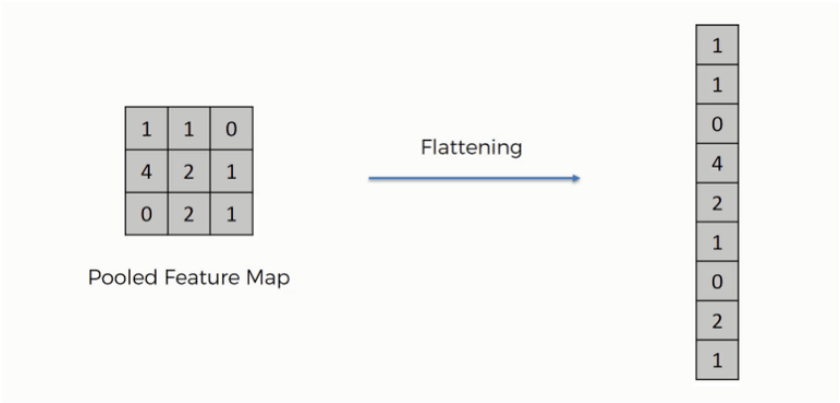
- **Max Pooling**: Thực hiện lấy giá trị lớn nhất trong kích thước filter mà ta xét.
- **Average Pooling**: Thực hiện lấy giá trị trung bình tổng trong kích thước filter ta xét.



Hình 2.5: Hoạt động của pooling

2.1.5 Lớp kết nối dày đặc

Trong phần phân lớp, ta sử dụng một vài tầng với kết nối đầy đủ - lớp kết nối dày đặc (fully connected layer) để xử lý kết quả của phần tích chập. Vì đầu vào của mạng liên kết đầy đủ là 1 chiều, ta cần làm phẳng đầu vào trước khi phân lớp (Hình 2.6). Tầng cuối cùng trong mạng CNN là một tầng liên kết đầy đủ, phần này hoạt động tương tự như mạng neural thông thường.

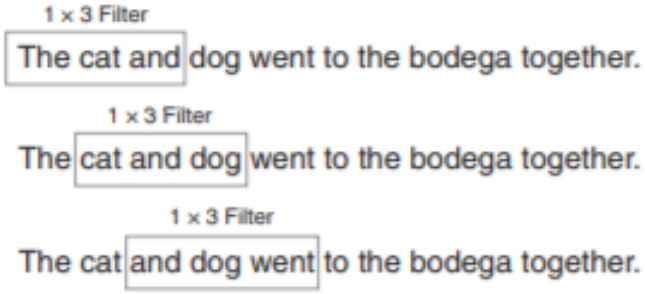


Hình 2.6: Làm phẳng đầu vào ảnh

Kết quả thu được cuối cùng cũng sẽ là một vector với các giá trị xác suất cho việc dự đoán như mạng neural thông thường.

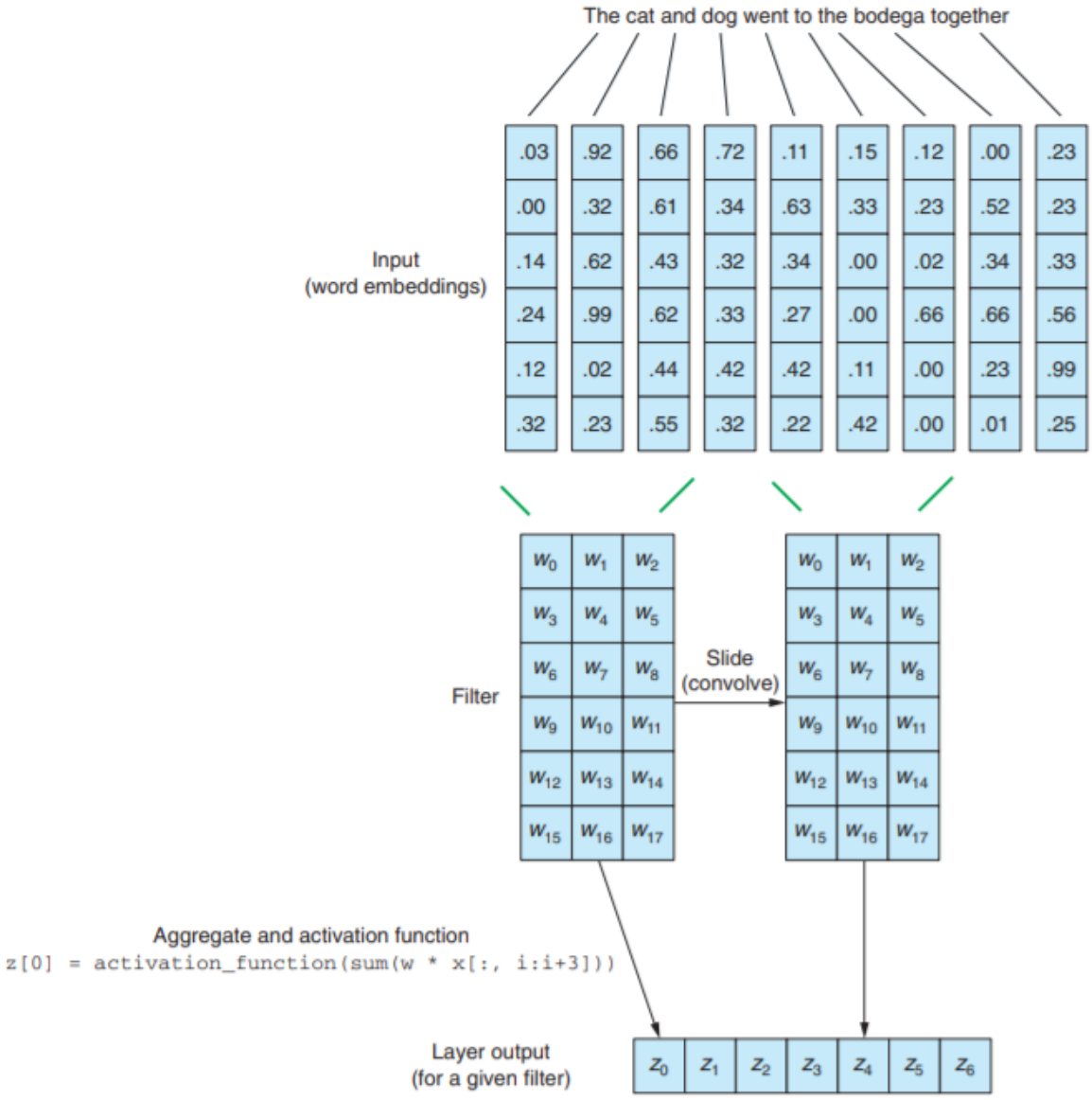
2.1.6 Ứng dụng mạng CNN trong NLP

Thuật toán CNN đã cho thấy sự thành công trong các bài toán phân loại ảnh. Và nó cũng áp dụng vào các bài toán của xử lý ngôn ngữ tự nhiên được. Trong ảnh, chúng ta áp dụng convolution 2D (2 chiều), còn trong NLP thì ý tưởng là áp dụng Conv1D.



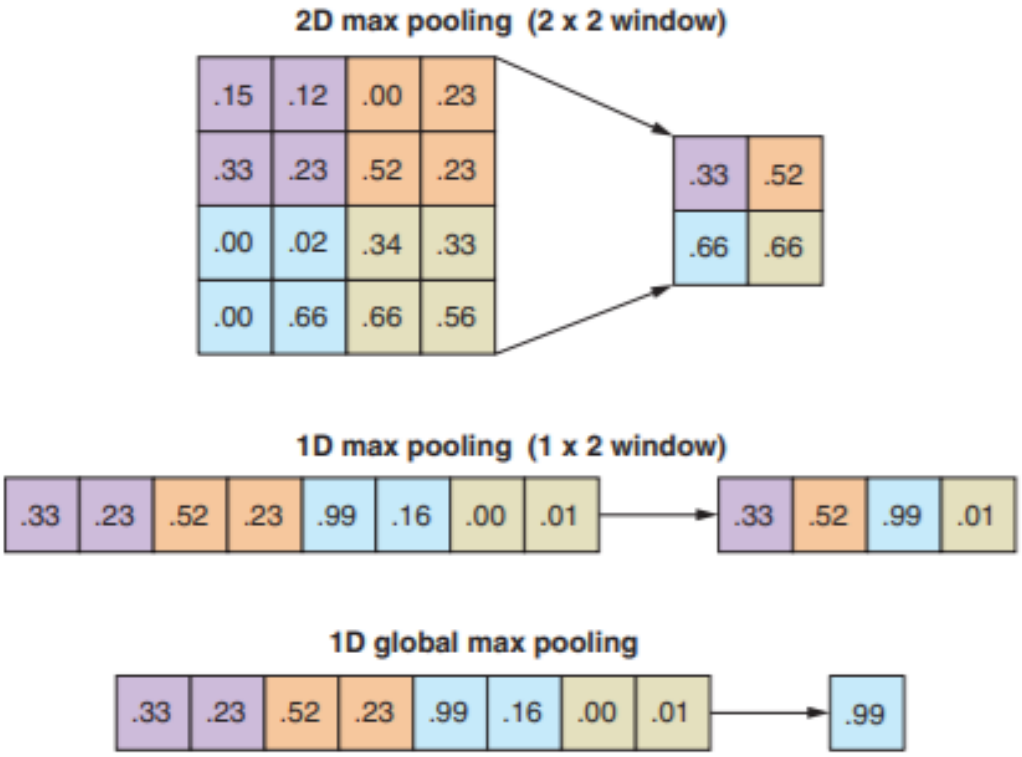
Hình 2.7: CNN trong NLP

Chúng ta sẽ coi văn bản như một hình ảnh, chiều cao coi như là chiều embedding. Chúng ta cần cố định chiều cao bộ lọc bằng với chiều embedding để thực hiện phép tích chập (Hình 2.8).



Hình 2.8: Trượt filter trên Word Embedding

Với pooling, ta có các loại sau (Hình 2.9):

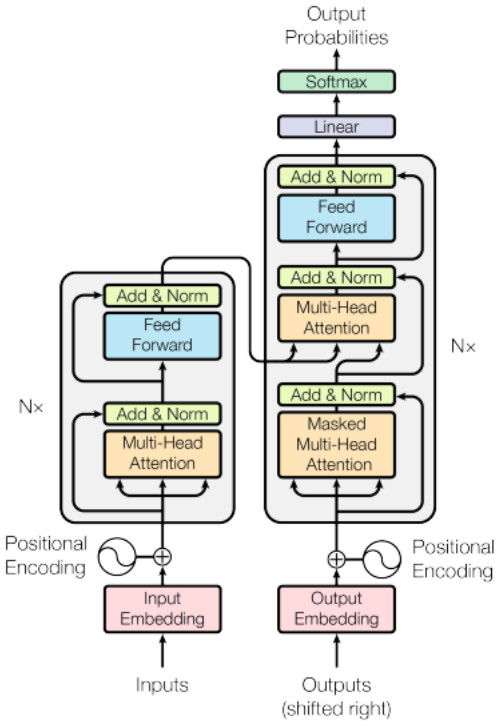


Hình 2.9: Pooling layer trong NLP

2.2 Mô hình PhoBert

2.2.1 Tổng quan về mô hình Transformer

Trong bài báo **Attention is all you need**, các tác giả giới thiệu về kiểu attention model được sử dụng trong các tác vụ học máy. Trong bài báo cũng đưa ra một kiến trúc mới là **Transformer** hoàn toàn khác so với các kiến trúc RNN trước đây, mặc dù cả hai đều thuộc lớp model seq2seq nhằm chuyển một câu văn đầu vào ở ngôn ngữ A sang 1 câu văn đầu ra ở ngôn ngữ B. Quá trình biến đổi (transorming) được dựa trên 2 phần encoder và decoder. Nghiên cứu chỉ ra rằng với cơ chế Attention không cần cần đến RNN đã có thể cải thiện hiệu quả của các tác vụ dịch máy và nhiều tác vụ khác.



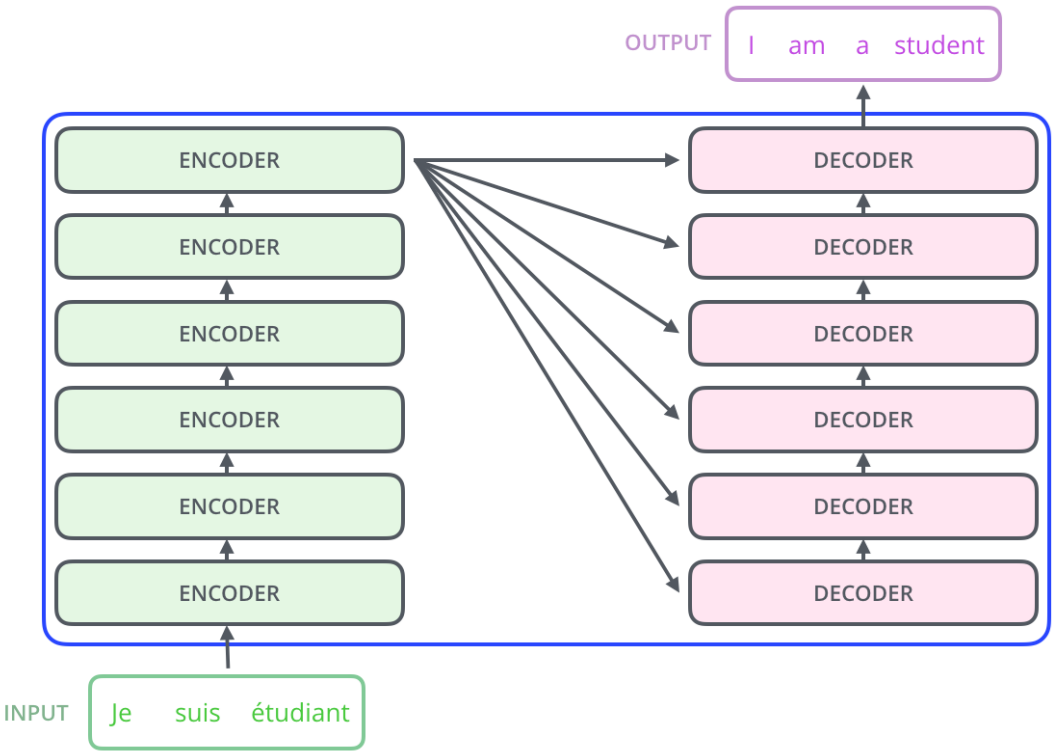
Hình 2.10: Kiến trúc mô hình Transformer

Kiến trúc mô hình **Transformer** gồm hai phân Encoder bên trái và Decoder bên phải (Hình 2.10).

- **Encoder**: Là tổng hợp xếp chồng lên nhau của 6 layers xác định. Mỗi layer bao gồm 2 layer con (sub-layer) trong nó. Sub-layer đầu tiên là multi-head attention. Layer thứ hai đơn thuần là các fully-connected feed-forward layer. Một lưu ý là chúng ta sẽ sử dụng một kết nối residual ở mỗi sub-layer, kiến trúc này tương tự như mạng resnet trong CNN. Đầu ra của mỗi sub-layer là *LayerNorm(x+Sublayer(x))* có số chiều là 512 như bài báo.

- **Decoder**: Cũng là tổng hợp của 6 layers xếp chồng nhau, kiến trúc tương tự như các sub-layer của Encoder ngoại trừ thêm một sub-layer thể hiện phân phối attention ở vị trí đầu tiên. Layer này không gì khác so với multi-head self-attention layer ngoại trừ được điều chỉnh để không đưa các từ trong tương lai vào attention. Tại bước thứ i của decoder chúng ta chỉ biết được các từ ở vị trí nhỏ hơn i nên việc điều chỉnh đảm bảo attention chỉ áp dụng cho những từ nhỏ hơn vị trí thứ i. Cơ chế residual cũng được áp dụng tương tự như trong Encoder.

Ngoài ra, chúng ta có một bước cộng thêm **Positional Encoding** vào các input của encoder và decoder nhằm thêm yếu tố thời gian vào mô hình để tăng độ chính xác. Đây đơn giản là phép cộng vector mã hóa vị trí của từ trong câu với vector biểu diễn từ. Chúng ta có thể mã hóa vị trí dưới dạng $\{0, 1\}$ vector vị trí hoặc sử dụng hàm *sin, cos* như trong bài báo.

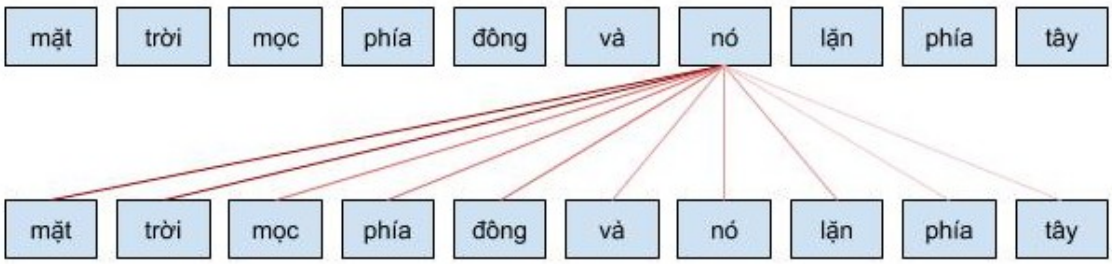


Hình 2.11: Kiến trúc mô hình Transformer sơ lược

Self-attention

Self Attention cho phép mô hình khi mã hóa một từ có thể sử dụng thông tin của những từ liên quan tới nó. Chúng ta có thể tưởng tượng cơ chế self attention giống như cơ chế tìm kiếm. Với một từ cho trước, cơ chế này sẽ cho phép mô hình tìm kiếm trong các từ còn lại, từ nào "giống" để sau đó thông tin sẽ được mã hóa dựa trên tất cả các từ trên.

Ví dụ chúng ta có câu: *mặt trời mọc phía đông và nó lặn ở phía tây* thì cơ chế selff attention sẽ giúp mô hình khi mã hóa từ *nó* thì sẽ tập trung vào từ liên quan là *mặt trời*.

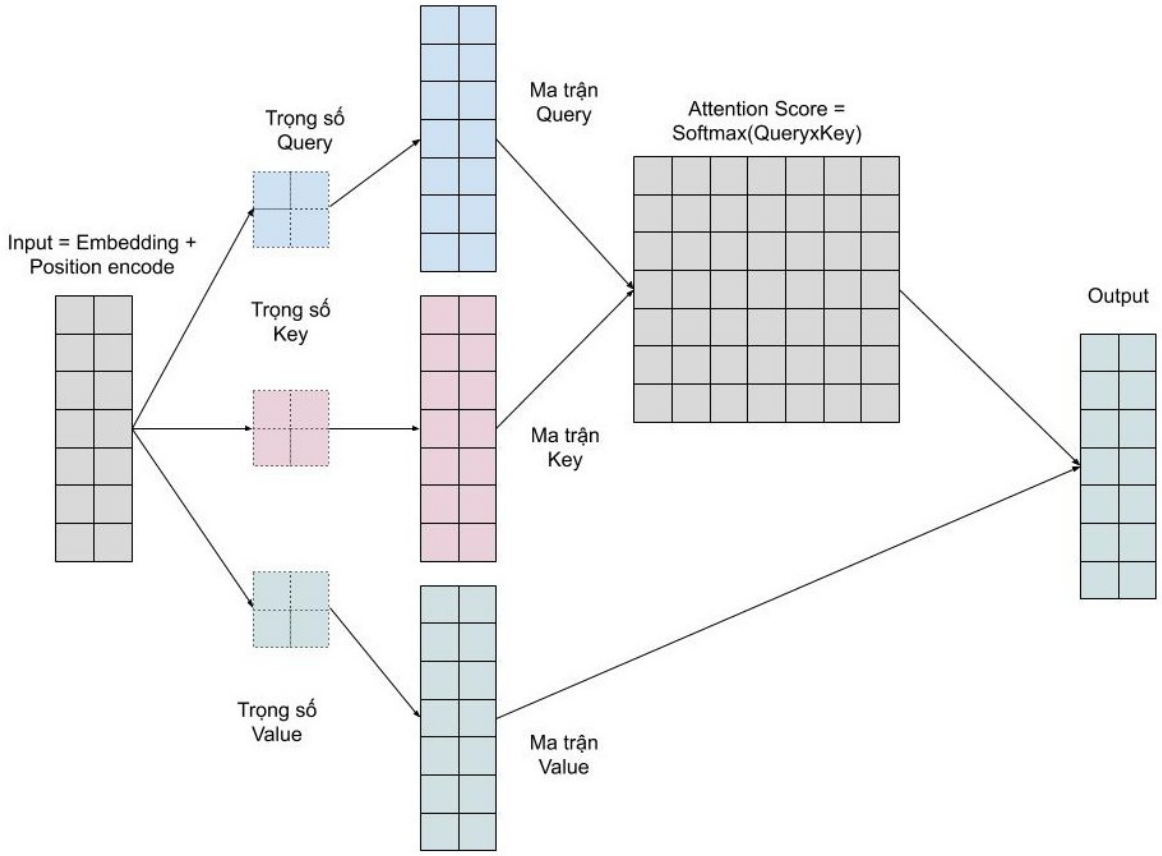


Hình 2.12: Ví dụ cơ chế self attention

X là ma trận đầu vào. X được chuyển thành các ma trận Query (Q), Key (K), Value (V) bằng cách nhân lần lượt với các ma trận trọng số W_q, W_k, W_v. Phương trình Attention như sau:

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

Trong đó d_k là số chiều key vector. Việc chia cho d_k nhằm tránh tràn luồng nếu số mũ là quá lớn.



Hình 2.13: Mô phỏng cách tính toán trên self attention

Multi-head Attention

Với self-attention, các tham số mà mô hình cần tinh chỉnh chính là các ma trận W_q, W_k, W_v. Mỗi quá trình như vậy được gọi là 1 head của attention. Khi lặp này quá trình này nhiều lần ta sẽ thu được quá trình **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. Công thức Multi-head Attention như sau:

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{concatenate}(head_1, head_2, ..., head_h)\mathbf{W_0}$$

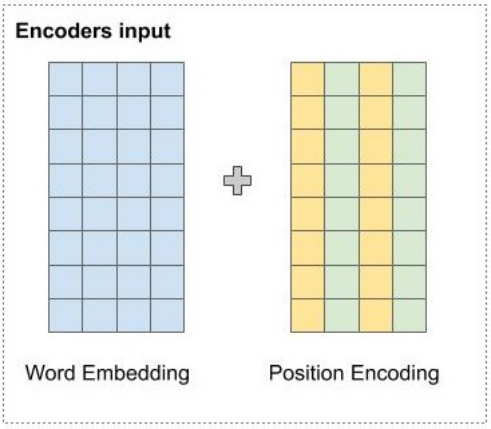
Ở đây $head_i = \text{Attention}(\mathbf{Q_i}, \mathbf{K_i}, \mathbf{V_i})$. Để trả về output có cùng kích thước với ma trận đầu vào chúng ta chỉ cần nhân với ma trận $\mathbf{W_0}$ chiều cao bằng với chiều rộng của ma trận đầu vào.

Position encoding

Transformer không dự báo tuần theo time step như RNN. Toàn bộ dữ liệu đầu vào sẽ được truyền thẳng vào mô hình cùng một thời điểm. Sẽ rất khó để nhận biết vị trí

của các từ đầu vào trong câu ở những lớp mô hình không tuân theo time step. Do đó thêm position encoding sẽ cho mô hình thêm các thông tin về vị trí của từ.

Position encoding vector sẽ được cộng trực tiếp vào embedding vector. Embeddings biểu diễn một token trong một không gian d chiều nơi mà các token có cùng ý nghĩa sẽ gần nhau hơn. Nhưng embedding vector không chứa thông tin vị trí của từ trong câu. Do đó sau khi thêm position encoding vector, một từ sẽ gần với những từ khác hơn dựa trên ý nghĩa của chúng và khoảng cách vị trí của chúng trong câu trong không gian d chiều.



Hình 2.14: Phương pháp Position Encoding

Cụ thể, tại vị trí chẵn, ta sử dụng hàm sin, và với vị trí lẻ tác giả sử dụng hàm cos để tính giá trị tại chiều đó.

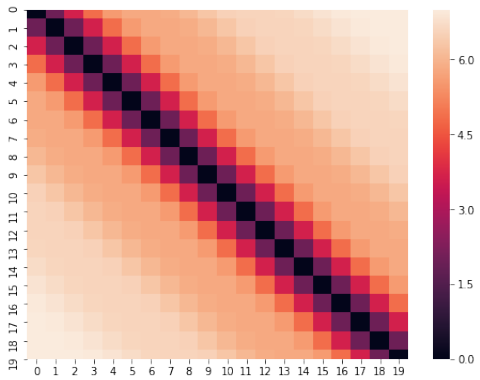
$$PE(pos, 2i) = sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$
$$PE(pos, 2i+1) = cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

Trong đó pos là vị trí hiện tại của từ, i là chỉ số của phần tử nằm trong vector encoding và d_{model} là kích thước các vector positional embedding. Vector này có kích thước bằng với các vector embedding từ để có thể thực hiện được phép cộng.

	0	1	2	3	4	5
0	$\sin(\frac{1}{1000^{0/512}} \times 0)$	$\cos(\frac{1}{1000^{0/512}} \times 0)$	$\sin(\frac{1}{1000^{2/512}} \times 0)$	$\cos(\frac{1}{1000^{2/512}} \times 0)$	$\sin(\frac{1}{1000^{4/512}} \times 0)$	$\cos(\frac{1}{1000^{4/512}} \times 0)$
1	$\sin(\frac{1}{1000^{0/512}} \times 1)$	$\cos(\frac{1}{1000^{0/512}} \times 1)$	$\sin(\frac{1}{1000^{2/512}} \times 1)$	$\cos(\frac{1}{1000^{2/512}} \times 1)$	$\sin(\frac{1}{1000^{4/512}} \times 1)$	$\cos(\frac{1}{1000^{4/512}} \times 1)$
2	$\sin(\frac{1}{1000^{0/512}} \times 2)$	$\cos(\frac{1}{1000^{0/512}} \times 2)$	$\sin(\frac{1}{1000^{2/512}} \times 2)$	$\cos(\frac{1}{1000^{2/512}} \times 2)$	$\sin(\frac{1}{1000^{4/512}} \times 2)$	$\cos(\frac{1}{1000^{4/512}} \times 2)$

Hình 2.15: Ví dụ Position Encoding với số chiều là 6

Hình 2.15 minh họa cho cách tính position encoding. Giả sử chúng ta có word embedding có 6 chiều, thì position encoding cũng có tương ứng là 6 chiều. Mỗi dòng tương ứng với một từ. Giá trị của các vector tại mỗi vị trí được tính toán theo công thức như hình.



Hình 2.16: Ma trận position encoding

Có thể thấy sau khi mã hóa, ma trận position encoding thu được các vector biểu diễn thể hiện được tính chất khoảng cách giữa 2 từ. 2 từ cách càng xa nhau thì khoảng cách càng lớn hơn.

2.2.2 Mô hình ngôn ngữ BERT

BERT là viết tắt của cụm từ **Bidirectional Encoder Respresetation from Transformer** là một mô hình được Google công bố vào tháng 11 năm 2018. Là mô hình biểu diễn từ theo 2 chiều ứng dụng kỹ thuật **Transformer**. BERT được thiết kế để huấn luyện trước các biểu diễn từ (pre-train word embedding) được sử dụng để transfer sang các bài toán khác trong lĩnh vực xử lý ngôn ngữ tự nhiên.

Cơ chế attention của Transformer sẽ truyền toàn bộ các từ trong câu văn đồng thời vào mô hình một lúc mà không cần quan tâm đến chiều của câu. Do đó Transformer được xem như là huấn luyện hai chiều (bidirectional) mặc dù trên thực tế chính xác hơn chúng ta có thể nói rằng đó là huấn luyện không chiều (non-directional). Đặc điểm này cho phép mô hình học được bối cảnh của từ dựa trên toàn bộ các từ xung quanh nó bao gồm cả từ bên trái và từ bên phải.

Kiến trúc BERT

Mô hình BERT là mô hình mã hóa sử dụng nhiều lớp Transformer hai chiều , trong đó các lớp Transformer được sử dụng hoàn toàn tương tự như mô hình được đề xuất và thực thi gốc của Vasawwani et al. (2017) . Kiến trúc của Transformer được nói rất kỹ ở phần trên và việc thực thi kiến trúc này trong BERT hoàn toàn giống với cách triển khai gốc đó.

Mô hình BERT định nghĩa số lượng lớp (khối Transformer) là **L**, kích thước của các lớp ẩn là **H** (hay còn gọi là kích thước của embedding vector) và số head ở lớp self attention là **A**. Tên gọi của 2 kiến trúc BERT bao gồm:

- $\textbf{BERT}_{\textbf{BASE}}(L = 12, H = 768, A = 12)$: Tổng tham số là 110 triệu.
- $\textbf{BERT}_{\textbf{LARGE}}(L = 24, H = 1024, A = 16)$: Tổng tham số là 340 triệu.

Biểu diễn dữ liệu đầu vào

Cách thức biểu diễn dữ liệu đầu vào của BERT cho phép biểu diễn đồng thời một câu hoặc một cặp câu (Ví dụ [câu hỏi, câu trả lời]) trong một chuỗi biểu diễn.

Khi có một chuỗi đầu vào cụ thể, biểu diễn đầu vào của chúng ta được xây dựng bằng cách tính tổng các token đó với vector phân đoạn và vị trí tương ứng của các từ trong chuỗi (Hình 2.17).

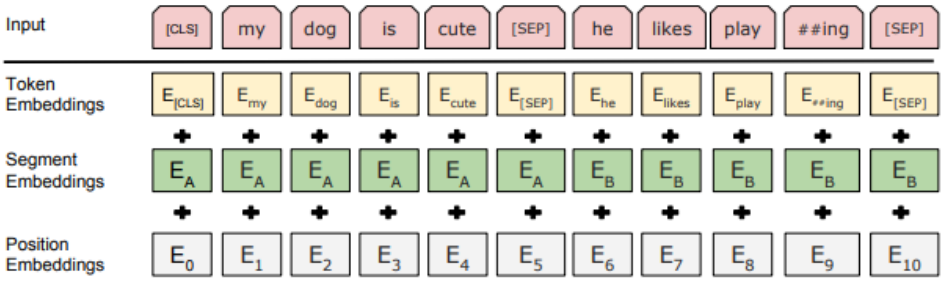


Figure 2: BERT input representation. The input embeddings is the sum of the token embeddings, the segmentation embeddings and the position embeddings.

Hình 2.17: Biểu diễn đầu vào của BERT

Một số điểm cần chú ý:

- BERT sử dụng bộ WordPiece (Wu et al.2016) để tách các câu thành các từ nhỏ với bộ từ điển bao gồm 30000 từ và sử dụng ## làm dấu phân tách. Ví dụ từ playing được tách thành play##ing.
- Lớp nhúng vị trí (positional embedding) được sử dụng với độ dài tối đa 512.
- Token đầu tiên của mõi chuỗi luôn mặc định là ký tự đăc biệt ([CLS]). Đầu ra của Transformer (hidden state cuối cùng) tương ứng với token này sẽ được sử dụng để đại diện cho cả câu trong các nhiệm vụ phân loại. Nếu không trong các nhiệm vụ phân loại, vector này được bỏ qua.
- Trong trường hợp cặp các câu được gộp lại với nhau thành một chuỗi duy nhất, chúng ta phân biệt các câu theo 2 cách:
  - Dấu hiệu thứ nhất là sử dụng ký tự ngăn cách [SEP].
  - Dấu hiệu thứ hai là sử dụng giá trị nhúng câu A đã được học cho chuỗi đầu tiên và giá trị nhúng B cho chuỗi thứ hai . Đối với tác vụ chỉ sử dụng một câu đơn , BERT chỉ sử sử dụng duy nhất giá trị nhứng A.
- Lớp Position Embeddings có nhiệm vụ mã hóa thông tin vị trí của từ trong câu. Nếu để ý , có thể thấy toàn bộ mô hình BERT chỉ sử dụng Transformer encoder tức là chỉ có attention và các mạng neural truyền thẳng với kỹ thuật normalize, hoàn toàn không hề có thứ tự như các mang neural hồi quy. Để mô hình có thể học được sự liên hệ về thứ tự xuất hiện của các từ thì người ta đưa thêm lớp Position Embeddings mang ý nghĩa về thứ tự của từ trong chuỗi.
- Lớp Segment Embedding mang thông tin về vị trí câu mà từ đó đang phụ thuộc. Ví dụ như hình những từ có Segment là E_A tức là nó thuộc câu A và ngược lại. Trên thực tế giá trị này sẽ là 0 nếu thuộc câu đầu tiên và 1 nếu thuộc câu thứ hai. Nếu tác vụ chỉ liên quan đến một câu thì chỉ cần gán các giá trị này bằng 0 hết.

- Lớp Token Embeddings có nhiệm vụ cho mô hình biết được vị trí của từ này trong tập từ điển, tương tự như mô hình word2vec nó được biểu diễn dưới dạng one hot vector với tập từ điển word piece 30000 token.

Masked Language Model

Mô hình BERT được tiền huấn luyện bởi hai tác vụ đó là **Mô hình ngôn ngữ đánh dấu - Masked Language Model** và **Dự đoán câu kế tiếp - Next Sentence Prediction**

Masked Language Model (gọi tắt là **MLM**) là mô hình mà bối cảnh của từ được học từ cả 2 phía bên trái và bên phải cùng một lúc từ những bộ dữ liệu văn bản không giám sát. Dữ liệu đầu vào sẽ được masked (thay bằng một token MASK) một cách ngẫu nhiên với tỷ lệ thấp. Huấn luyện mô hình dự báo được từ masked dựa trên bối cảnh xung quanh là những từ không được masked nhằm tìm ra cách biểu diễn của từ.

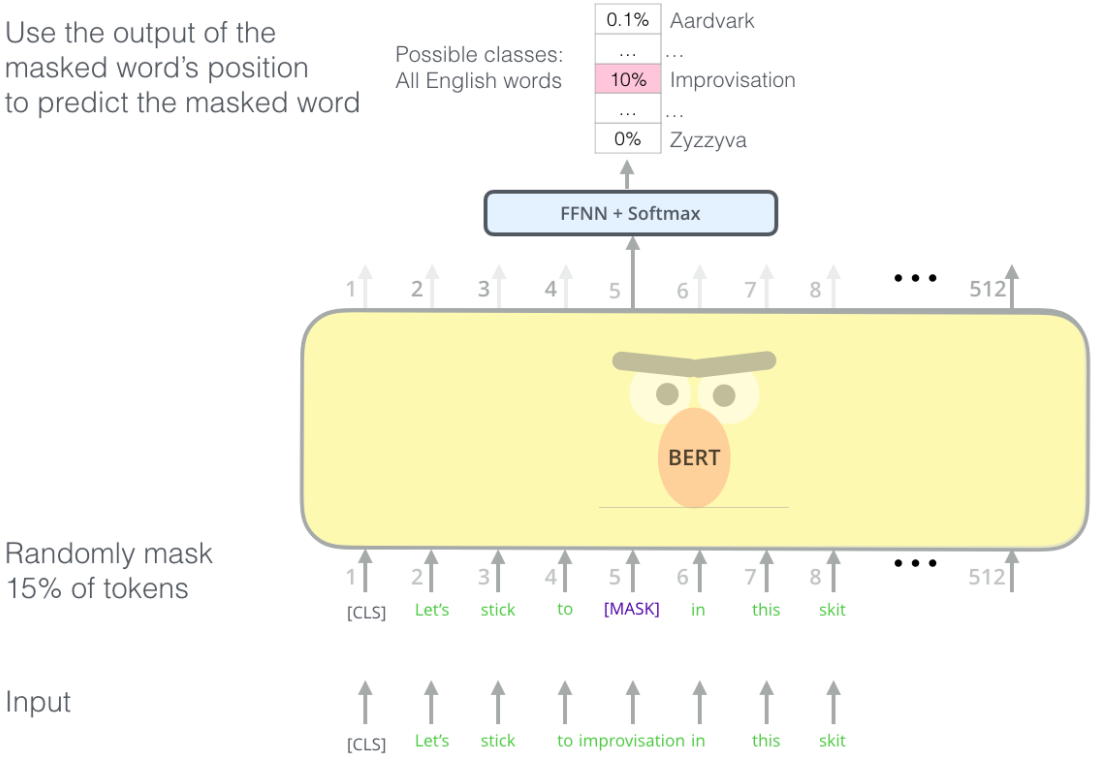
Khoảng 15% các token của câu input được thay thế bởi [MASK] token trước khi truyền vào model đại diện cho những từ bị che dấu (masked). Mô hình sẽ dựa trên các từ không được che (non-masked) dấu xung quanh [MASK] và đồng thời là bối cảnh của [MASK] để dự báo giá trị gốc của từ được che dấu. Số lượng từ được che dấu được lựa chọn là một số ít (15%) để tỷ lệ bối cảnh chiếm nhiều hơn (85%).

Cách tiếp cận này vẫn tồn tại hai nhược điểm sau khi nhận được mô hình tiền huấn luyện theo hai chiều . Nhược điểm thứ nhất là BERT tạo ra sự không phù hợp giữa quá trình tiền huấn luyện và quá trình tinh chỉnh mô hình , tức là ký tự [MASK] sẽ không bao giờ xuất hiện trong quá trình tinh chỉnh mô hình . Để giảm thiểu vấn đề này , BERT không luôn luôn đánh dấu bằng ký tự [MASK]. Thay vào đó:

- Thay thế 80% số lượng từ được chọn bằng [MASK].
- Thay thế 10% số lượng từ được chọn bằng một từ bất kỳ khác.
- Giữ nguyên 10% số lượng từ được chọn.

Hàm loss function của BERT sẽ bỏ qua mất mát từ những từ không bị che dấu và chỉ đưa vào mất mát của những từ bị che dấu. Do đó mô hình sẽ hội tụ lâu hơn nhưng đây là đặc tính bù trừ cho sự gia tăng ý thức về bối cảnh. Lớp Tranformer encoder không hề biết trong tập hợp các từ được chọn, từ nào sẽ bị thay thế bằng những từ khác và từ nào sẽ giữ nguyên, do đó mô hình sẽ học được phân phối của tất cả các từ. Thêm vào đó, bởi vì việc thay thế ngẫu nhiên chỉ xảy ra cho 1.5% số lượng các từ nên có vẻ sẽ không ảnh hưởng đến tính phù hợp của việc thu nhận kiến thức cho mô hình ngôn ngữ.

Nhược điểm thứ hai là việc lựa chọn ngẫu nhiên 15% số lượng các từ bị che dấu cũng tạo ra vô số các kịch bản input cho mô hình huấn luyện nên mô hình sẽ cần phải huấn luyện rất lâu mới học được toàn diện các khả năng.



Hình 2.18: BERT cho tác vụ Masked Language Model

Next Sentence Prediction

Đây là một bài toán phân loại học có giám sát với 2 nhãn (hay còn gọi là phân loại nhị phân). Input đầu vào của mô hình là một cặp câu (pair-sequence) sao cho 50% câu thứ 2 được lựa chọn là câu tiếp theo của câu thứ nhất và 50% được lựa chọn một cách ngẫu nhiên từ bộ văn bản mà không có mối liên hệ gì với câu thứ nhất. Nhãn của mô hình sẽ tương ứng với **IsNext** khi cặp câu là liên tiếp hoặc **NotNext** nếu cặp câu không liên tiếp.

Cũng tương tự như mô hình Question and Answering, chúng ta cần đánh dấu các vị trí đầu câu thứ nhất bằng token [CLS] và vị trí cuối các câu bằng token [SEP]. Các token này có tác dụng nhận biết các vị trí bắt đầu và kết thúc của từng câu thứ nhất và thứ hai.

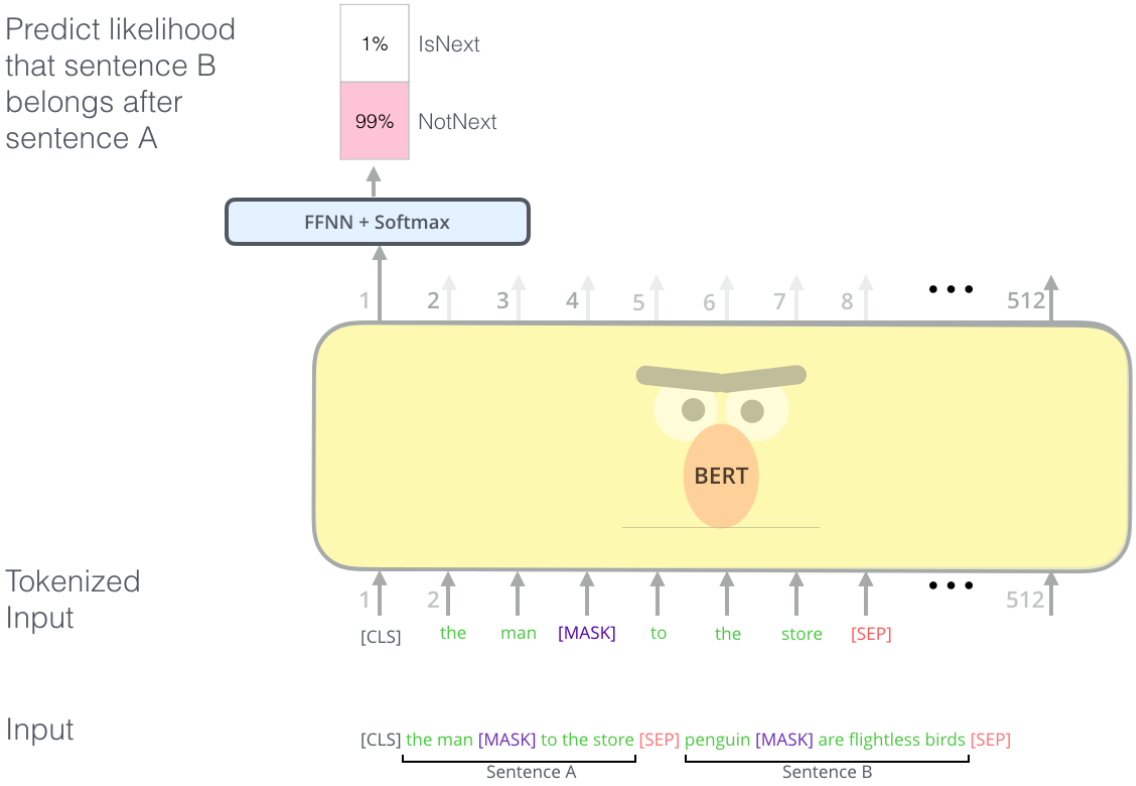
Ví dụ chúng ta có các đầu vào và nhãn như sau:

Input: [CLS] người đàn_ông làm [MASK] tại cửa_hàng [SEP] anh_ta rất [MASK] và thân_thiện [SEP]

Label: isNext

Input: [CLS] người đàn_ông làm [MASK] tại cửa_hàng [SEP] cô_ta đang cầm súng [SEP]

Label: notNext

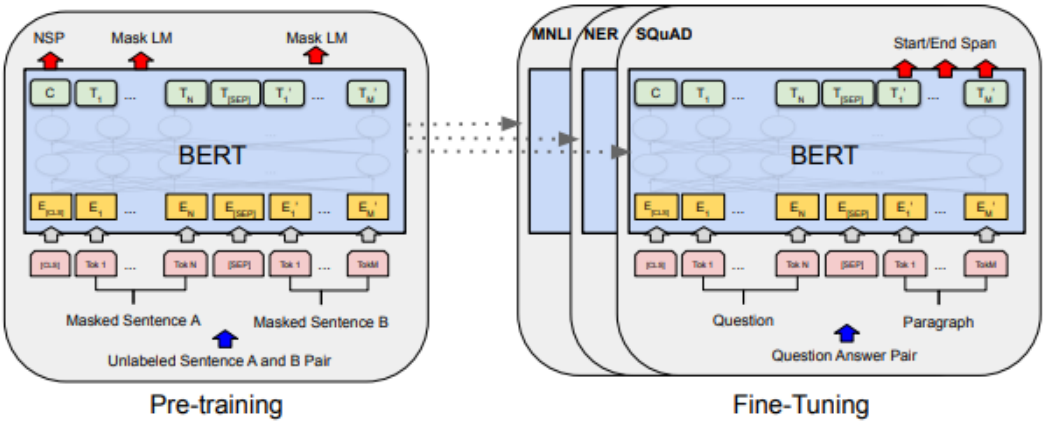


Hình 2.19: BERT cho tác vụ Next sentence prediction

Chúng ta chọn những câu notNext một cách ngẫu nhiên và mô hình cuối cùng đạt được độ chính xác 97%-98% trong nhiệm vụ này.

Fine-tuning BERT

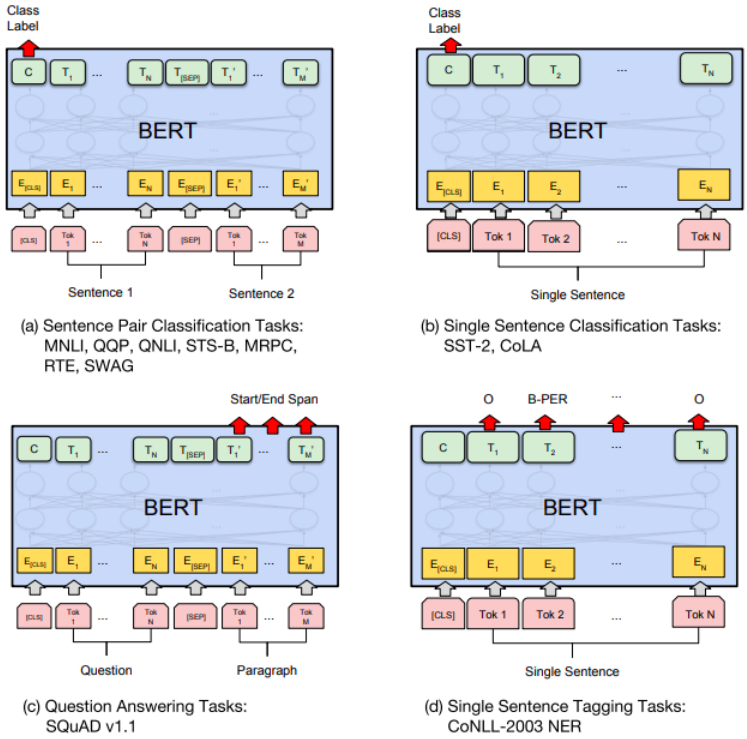
Một điểm đặc biệt ở BERT mà các model embedding trước đây chưa từng có đó là kết quả huấn luyện có thể fine-tuning được. Chúng ta sẽ thêm vào kiến trúc model một output layer để tùy biến theo tác vụ huấn luyện.



Hình 2.20: Tiến trình pre-train và fine-tuning của BERT

Trong suốt quá trình fine-tuning thì toàn bộ các tham số của layers học chuyển giao sẽ được fine-tune. Đối với các tác vụ sử dụng input là một cặp sequence (pair-sequence) ví dụ như question and answering thì ta sẽ thêm token khởi tạo là [CLS] ở đầu câu, token [SEP] ở giữa để ngăn cách 2 câu. Tiến trình áp dụng fine-tuning sẽ như sau:

- **Bước 1:** Embedding toàn bộ các token của cặp câu bằng các véc tơ nhúng từ pretrain model. Các token embedding bao gồm cả 2 token là [CLS] và [SEP] để đánh dấu vị trí bắt đầu của câu hỏi và vị trí ngăn cách giữa 2 câu. 2 token này sẽ được dự báo ở output để xác định các phần Start/End Spand của câu output.
- **Bước 2:** Các embedding véc tơ sau đó sẽ được truyền vào kiến trúc multi-head attention với nhiều block code (thường là 6, 12 hoặc 24 blocks tùy theo kiến trúc BERT). Ta thu được một véc tơ output ở encoder.
- **Bước 3:** Để dự báo phân phối xác suất cho từng vị trí từ ở decoder, ở mỗi time step chúng ta sẽ truyền vào decoder véc tơ output của encoder và véc tơ embedding input của decoder để tính encoder-decoder attention. Sau đó projection qua liner layer và softmax để thu được phân phối xác suất cho output tương ứng ở time step t.
- **Bước 4:** Trong kết quả trả ra ở output của transformer ta sẽ cố định kết quả của câu Question sao cho trùng với câu Question ở input. Các vị trí còn lại sẽ là thành phần mở rộng Start/End Span tương ứng với câu trả lời tìm được từ câu input.



Hình 2.21: Mô hình BERT với các tác vụ khác nhau

Lưu ý quá trình huấn luyện chúng ta sẽ fine-tune lại toàn bộ các tham số của model BERT đã cut off top linear layer và huấn luyện lại từ đầu các tham số của linear layer mà chúng ta thêm vào kiến trúc model BERT để customize lại phù hợp với bài toán.

BERT là một khái niệm đơn giản nhưng lại mang lại hiệu quả cực lớn trong thực tế. Nó đã thu được kết quả tối ưu mới nhất cho 11 nhiệm vụ xử lý ngôn ngữ tự nhiên, bao gồm:

- Tăng GLUE score (General Language Understanding Evaluation score), một chỉ số tổng quát đánh giá mức độ hiểu ngôn ngữ lên 80.5%.
- Tăng accuracy trên bộ dữ liệu MultiNLI đánh giá tác vụ quan hệ văn bản (text entailment) lên 86.7%.
- Tăng accuracy F1 score trên bộ dữ liệu SQuAD v1.1 đánh giá tác vụ question and answering lên 93.2%.

2.2.3 PhoBert

PhoBert là mô hình dựa trên kiến trúc của **RoBerta**, nhưng được huấn luyện và xây dựng trên miền tiếng Việt. PhoBert được huấn luyện trên kho dữ liệu Vietnamese Wikipedia (1GB) và kho dữ liệu thứ hai (19GB) là được tạo ra bằng cách xóa các bài báo tương tự và trùng lặp khỏi kho tin tức tiếng Việt 50GB. PhoBert sử dụng fastBPE để phân đoạn câu thành các subword. Sử dụng bộ từ vựng gồm 64K subword. Trung bình có 24,4 token subword cho mỗi câu.

RoBERTa là viết tắt của cụm từ **Robustly optimized BERT approach** được giới thiệu bởi Facebook là một phiên bản được huấn luyện lại BERT với một phương pháp huấn luyện tốt hơn với dữ liệu được tăng gấp 10 lần.

Để tăng cường quá trình huấn luyện, RoBERTa không sử dụng cơ chế dự đoán câu kế tiếp (NSP) từ BERT mà sử dụng kỹ thuật mặt nạ động (dynamic masking), theo đó các token mặt nạ sẽ bị thay đổi trong quá trình huấn luyện. Sử dụng kích thước batch lớn hơn cho thấy hiệu quả tốt hơn khi huấn luyện.

Một điều quan trọng nữa, RoBERTa sử dụng 160GB văn bản để huấn luyện. Trong đó, 16GB là sách và Wikipedia tiếng Anh được sử dụng trong huấn luyện BERT. Phần còn lại bao gồm CommonCrawl News dataset (63 triệu bản tin, 76 GB), ngữ liệu văn bản Web (38 GB) và Common Crawl Stories (31 GB). Mô hình này được huấn luyện với GPU của Tesala 1024 V100 trong một ngày. Huấn luyện mô hình lâu hơn, với batch size lớn hơn và trên nhiều dữ liệu hơn.

Roberta cùng kiến trúc với BERT nhưng sử dụng mã hóa BPE để làm tokenizer. Robert không sử dụng Segment Embedding, chúng ta chỉ cần chỉ ra token thuộc phân đoạn nào bằng cách sử dụng token </s>.

2.3 Mô hình LightGBM

2.3.1 Boosting

Các giải thuật Boosting đều thuộc hệ thống kỹ thuật ensemble, ý tưởng chung đều là kết hợp nhiều mô hình học yếu thành một mô hình học có hiệu năng tốt hơn cho từng mô hình đơn lẻ. Cụ thể, kỹ thuật ensemble là một khái niệm trong học máy, thực hiện huấn luyện đa mô hình học với cùng một thuật toán.

Những yếu tố gây ra lỗi trong quá trình huấn luyện mô hình là bias và variance. Kỹ thuật ensemble giúp giảm thiểu những lỗi đó. Kỹ thuật Ensemble giúp giảm thiểu những yếu tố đó. Kỹ thuật này được thiết kế nhằm cải thiện sự ổn định và độ chính sách của các giải thuật Học Máy. Kết hợp nhiều bộ phân loại làm giảm lỗi do variance, đặc biệt lỗi do bộ phân loại không ổn định, và có thể tạo ra bộ phân loại đáng tin cậy hơn so với một bộ phân loại đơn. Thông thường trong Boosting ta phải chọn một giải thuật học nền. Ví dụ, nếu ta chọn giải thuật Decision Tree, kỹ thuật Boosting sẽ là kết hợp của tập các cây với số lượng tùy vào ta chọn.

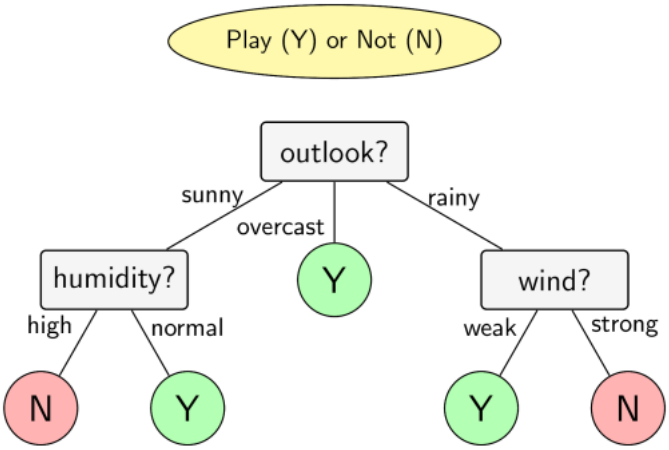
Giả sử cần huấn luyện N bộ phân loại $(n_0, n_1, \ldots, n_N)$, lần lượt các mô hình sẽ huấn luyện từ n_0 đến n_N. Bộ phân loại n_i được huấn luyện từ trạng thái hội tụ của bộ phân loại thứ n_{i-1}. Như vậy, quá trình huấn luyện có N state tương ứng với N bộ phân loại. Tại mỗi stage tập dữ liệu training Bộ phân loại tại stage đó sẽ được sample ngẫu nhiên từ tập training gốc. Thông thường, để tăng hiệu năng phân loại sau mỗi stage các điểm dữ liệu sẽ được gán lại trọng số, giá trị trọng số thường tỉ lệ thuận với giá trị lỗi do điểm dữ liệu đó gây ra trong stage trước đó. Bằng cách này khi training Bộ phân loại của stage mới sẽ chú trọng vào tối ưu cho những điểm có trọng số gây lỗi cao này. Ngoài ra, tại mỗi stage tùy vào hiệu năng phân loại của Bô phân loại của stage đó mà bộ phân loại này sẽ được gán với một trọng số. Ví dụ, tại state i, bộ phân loại n_i được gán trọng số w_i, khi đó, quá trình phân loại cho một điểm dữ liệu mới sẽ là:

$$o = \underset{i=0}{N}(w_i * o_i)$$

Trong đó, o_i là kết quả phân loại của bộ phân loại thứ i.

2.3.2 Gradient Boosting Decision Tree

Giải thuật Gradient Boosting Decision Tree (GBDT) là một giải thuật thuộc lớp giải thuật Boosting, với giải thuật gốc là Decision Tree. Decision Tree thường được áp dụng cho bài toán phân loại (hình 2.22).



Hình 2.22: Minh họa Decision Tree

Trên cây có lá là một phân bố của các nhãn phân loại, bên trong các node là một feature đầu vào của điểm dữ liệu. Quá trình phân loại sẽ là duyệt từ gốc và dựa vào feature của node đang xét và giá trị feature đó của điểm dữ liệu để rẽ nhánh đến khi gặp nút lá. Trong giải thuật Decision Tree sẽ có những tham số cần tối ưu như, độ sâu của cây, giá trị tối thiểu ở mỗi node để có thể rẽ nhánh, ... Trong GDBT, các tham số của các Cây, các trọng số của giải thuật Boosting đều được tối ưu sử dụng Gradient Descent. Bằng cách làm này, kết chọn một số lượng cây phù hợp ta có thể có được một model phân loại rất tốt và đảm bảo tốc độ phân loại.

2.3.3 LightGBM

Gradient boosting decision tree (GBDT) được sử dụng rộng rãi trong học máy với độ chính xác và hiệu quả cao, thường được áp dụng cho bài toán phân loại đa nhãn lớp. Tuy nhiên cùng với sự xuất hiện của dữ liệu lớn cả về số lượng và số chiều là một thách thức đối với GBDT. Để giải quyết vấn đề này thì LightGBM sử dụng 2 kỹ thuật là: Gradient-based One-Side Sampling để giảm số lượng dữ liệu và Exclusive Feature Bundling để giảm chiều của dữ liệu

Gradient-based One-Side Sampling

Gradient-based One-Side Sampling (GOSS) là một kỹ thuật giảm số lượng dữ liệu của tập dữ liệu. Giải thuật GBDT thông thường thì các sample không có trọng số (weight), như vậy sẽ không thể đánh giá được mức độ quan trọng của mỗi sample trong tập samples. Mỗi sample có gradient khác nhau sẽ đóng góp vai trò khác nhau trong việc tính toán lượng thông tin giữ lại của tập samples. Theo định nghĩa của lượng thông tin được giữ lại (information gain) thì những sample có gradient lớn sẽ đóng góp nhiều hơn đối với lượng thông tin được dữ lại. Vì thế khi giảm số lượng sample thì

chúng ta sẽ ưu tiên giữ lại những samples có gradient lớn và loại bỏ ngẫu nhiên một số lượng nhất định những sample có gradient nhỏ.

GOSS đầu tiên sắp xếp những sample theo gradient của chúng và lựa chọn lượng sample có gradient cao. Sau đó chọn ngẫu nhiên sample có gradient thấp. Sau đó khuếch đại tập dữ liệu cùng với gradient thấp với một hằng số $\frac{1-a}{b}$ khi tính toán lượng thông tin dữ lại. Việc nhân với một hằng số này để không làm thay đổi nhiều phân phối của tập dữ liệu gốc.
Dưới đây là mã giả của GOSS:

Algorithm 1: *GOSS*, Gradient-based One-Side Sampling

Input: I: training data, d: iterations ;
Input: a: sampling ratio of large gradient data ;
Input: b: sampling ratio of small gradient data ;
Input: $loss$: loss function, L: weak learner ;
models $\leftarrow \{\}$, fact $\leftarrow \frac{1-a}{b}$;
topN $\leftarrow a \times len(I)$, randN $\leftarrow b \times len(I)$;
for *i=1* ***to*** d **do**
    preds $\leftarrow$ models.predict(I) ;
    g $\leftarrow loss$(I, preds), w $\leftarrow$ {1,1,...} ;
    sorted $\leftarrow$ GetSortedIndices(abs(g)) ;
    topSet $\leftarrow$ sorted[1:topN] ;
    randSet $\leftarrow$ RandomPick(sorted[topN:len(I)], randN) ;
    usedSet $\leftarrow$ topSet + randSet ;
    w[randSet] $\times =$ fact (Assign weight $fact$ to the small gradient data) ;
    newModel $\leftarrow$ L(I[usedSet], - g[usedSet], w[usedSet]) ;
    models.append(newModel) ;
end

Exclusive Feature Bundling

Exclusive Feature Bundling (EFB) là một kỹ thuật để giảm số chiều của dữ liệu. Kỹ thuật này đặc biệt hiệu quả với dữ liệu thưa, dữ liệu với số chiều lớn. Đặc biệt trong không gian dữ liệu thưa có một vài thuộc tính (features) là exclusive, tức là chúng hiếm khi cùng nhận giá trị 0 đồng thời. Ví dụ như one-hot features, one-hot word trong khai phá văn bản. Chúng ta có thể bó các features này lại với nhau. Việc lựa chọn các thuộc tính nào được bó lại với nhau thì ta đi giải quyết bằng bài toán tô màu đồ thị (bằng cách lấy các features như những đỉnh và thêm nhưng cạnh cho 2 feature nếu chúng không loại trừ lẫn nhau. Và giải quyết bài toán này bằng một thuật toán tham lam cùng 1 tỷ lệ xấp xỉ không đổi.
Dưới đây là 2 thuật toán sử dụng trong kỹ thuật EFB.

Algorithm 2: Greedy Bundling

Input: F: features, K: max conflict count ;
Construct graph G ;
searchOrder ← G.sortByDegree() ;
bundles ← {}, bundlesConflict ← {} ;
for *i **in** searchOrder* **do**
    needNew ← True ;
end
for *j-1 **to** len(bundles)* **do**
    cnt ← ConflictCnt(bundles[j], F[i]) ;
    if *cnt+bundlesConflictCnt[i]* $\leq K$ **then**
        bundles[j].add(F[i]), needNew ← False ;
        break ;
    end
    .
end
if *needNew* **then**
    Add $F[i]$ as a new bundle to *bundles* ;
end
. **Output:** *bundles* ;

Algorithm 3: Merge Exclusive Features

Input: *numData*: number of data ;
Input: F: One bundle of exclusive features ;
binRanges ← {0}, totalBin ← 0 ;
for f ***in*** F **do**
    totalBin += f.numBin ;
    binRanges.append(totalBin) ;
end
newBin ← new Bin(numData) **for** $i = 1$ ***to*** *numData* **do**
    newBin[i] ← 0 ;
    for $j = 1$ ***to*** $len(F)$ **do**
        if $F[j].bin[i] \neq 0$ **then**
            newBin[i] ← F[j].bin[i] + binRanges[j] ;
        end
    end
end
Output: newBin, binRanges ;

Dữ liệu với số chiều lớn luôn luôn là thưa. Sự thưa thớt của không gian feature cung cấp cho ta một cách tiếp cận gần như không mất mát để giảm số lượng features. Chúng ta có thể bó những features exclusive này lại thành 1 feature. Bằng một thuật toán scan những tính năng được thiết kế cẩn thận, chúng ta có thể xây dựng được histograms của những feature giống nhau từ các bundles feature như những feature

riêng lẻ. Với cách xây dựng này có chi phí thay đổi từ $O(\#data \times \#feature)$ xuống $O(\#data \times \#bundle)$ với $\#bundle << \#feature$.

Ở đây có 2 vấn đề cần giải quyết là : Xác định những feature nào sẽ được bó cùng nhau (Algorithm 2) và cách xây dựng các bundle (Algorithm 3). Theo LightGBM thì ta có định lý sau: "Bài toán phân chia những feature đến một số lượng nhỏ hơn của những exclusive bundles là bài toán NP-hard".

Đối với vấn đề thứ nhất: xác định các features nào sẽ được bó cùng nhau. Thì theo định lý NP-hard đã chứng minh là NP-hard là tối ưu để tìm ra chiến lược bó. Và nó cũng chỉ ra rằng không thể tìm ra một giải pháp chính xác trong thời gian đa thức. Để tìm được một thuật toán xấp xỉ tốt thì ta phải chuyển từ bài toán tối ưu việc bó các features thành bài toán tô màu đồ thị trong đó các features là các đỉnh, thêm các cạnh nếu 2 feature không loại trừ lẫn nhau. Sau đó chúng ta sử dụng 1 thuật toán tham lam để tạo ra một kết quả khá tốt với tỉ lệ xấp xỉ không đổi để tô màu cho đồ thì giúp tạo ra các bundles.

Đối với vấn đề thứ 2: Xây dựng các bundles. Sau khi chúng ta xác định được các features nào sẽ được bó với nhau vào cùng một bundle thì ta cần một cách tối ưu nhất để gộp (merging) vào cùng một bundle để giảm độ phức tạp trong quá trình training. Điều quan trọng là cần phải đảm bảo rằng giá trị trong feature gốc có thể được xác định từ những feature đã được bó (bundles). Do thuật toán dựa trên histogram lưu trữ những bins rời rạc thay vì những giá trị liên tục nên chúng ta có thể xây dựng 1 feature bundle bằng cách để các exclusive features trong các bin khác nhau. Điều này có thể đạt được bằng cách thêm các offsets vào những giá trị ban đầu của các features. Ví dụ: giả sử ta có 2 features (A, B) trong 1 feature bundle. Ban đầu feature A có giá trị từ [0,10) và feature B có giá trị [0,20). Sau đó ta thêm 1 offset của 10 đến tất cả giá trị của B vì vậy B nằm trong khoảng từ [10,30). Sau khi merge 2 feature A và B ta được 1 feature bundle có giá trị từ [0, 30] để thay thế cho 2 feature ban đầu là A và B.

Như vậy EFB có thể kết hợp các exclusive features thành 1 feature bundle, qua đó có thể tránh những tính toán không cần thiết cho những giá trị bằng 0.

Chương 3

Thực nghiệm

3.1 Xây dựng bộ dữ liệu

Để chuẩn bị cho quá trình nghiên cứu, thử nghiệm và triển khai ứng dụng cần phải có một bộ dữ liệu sử dụng cho quá trình huấn luyện mạng, trích rút đặc trưng; xây dựng mạng học máy, học sâu và sau đó sử dụng mô hình thích hợp vào ứng dụng thực tiễn. Dữ liệu cũng quyết định rất lớn đến kết quả. Vì vậy, bước xây dựng dữ liệu là bước quan trọng và cũng đòi hỏi nhiều thời gian và công sức nhất.

3.1.1 Crawl dữ liệu

Để có một bộ dữ liệu đa dạng, em thực hiện crawl dữ liệu trên internet. Các nguồn phổ biến là các trang hỏi đáp về lĩnh vực y tế được người dùng đặt câu hỏi và các chuyên gia, bác sĩ đã trả lời. Những dữ liệu thô này sau đó được xử lý và đẩy vào hệ thống gán nhãn để thực hiện gán nhãn câu hỏi tương đồng và đồng thời kiểm tra lại tính chính xác của câu hỏi và câu trả lời.

Trong phần này, thư viện **Scrapy** của Python được sử dụng để hỗ trợ crawl dữ liệu. Scrapy là một thư viện mã nguồn mở miễn phí được viết bằng Python (chi tiết tại `https://scrapy.org/`). Ban đầu được thiết kế để tìm kiếm trên web, nó cũng có thể được sử dụng để trích xuất dữ liệu sử dụng API hoặc như một trình thu thập thông tin web. Nó giúp xây dựng và mở rộng dễ dàng các dự án thu thập dữ liệu . Quá trình thu thập dữ liệu này thu được hơn 9000 bộ câu hỏi và câu trả lời tương ứng về lĩnh vực y tế. Dữ liệu được lưu trữ lại vào file định dạng csv với 2 cột là *question* và *answer* tương ứng là các câu hỏi và câu trả lời.

question	answer
Chỉ số xét nghiệm ca 125 là gì?	Xét nghiệm CA 125 chính là chỉ số kháng nguyên ung th...
3 món ăn đơn giản giúp ngực to một cách tự nhiên	Có thể bạn không biết những món ăn đơn giản thường...
Giá khám tổng quát định kì	Như các bạn biết thì chi phí khám sức khỏe tổng quát t...
Đừng để bị giảm thị lực vĩnh viễn do viêm bờ mi	Bạn có thể gặp nhiều phiền toái, mất thẩm mĩ vùng mắt...
5 thời điểm vàng bạn nên uống mật ong	Bạn dùng mật ong vào buổi sáng để lọc sạch cặn bẩn sa...
Thông tin mới vụ hàng chục trẻ bị sùi mào gà sau cắt ba...	Chiều 17. 7, thanh tra Sở Y tế Hưng Yên cho biết, phòng...
Dấu hiệu ung thư vòm họng giai đoạn cuối	Ung thư, là căn bệnh gây tử vong nhiều nhất trong các l...
Trước khi làm xét nghiệm máu có nên cho bé uống sữa...	Đây là vấn đề được rất nhiều bà mẹ quan tâm bởi việc...
Bệnh viêm màng não có tỉ lệ di chứng cao	Viêm màng não là bệnh nguy hiểm, dẫn đến những hậu...
Mẹo để đồ ăn nhanh lành mạnh hơn với trẻ	Bạn cảm thấy khó khăn trong việc cung cấp những bữa...
Những loại quả ngon cho mẹ bầu 3 tháng thèm chua	Khi mang thai, khẩu vị của mẹ bầu cũng vô cùng đa dạn...
Phân biệt sùi mào gà với mụn rộp sinh dục	Bệnh sùi mào gà và mụn rộp sinh dục do có nhiều triệu...
Di chứng nguy hiểm khi trẻ sơ sinh bị ngạt do sặc nước ối	Làm mẹ là thiên chức của người phụ nữ, và cũng là niề...
5 cách đơn giản để thanh lọc phổi	Phổi tích tụ nhiều chất độc có thể gây ra nhiều vấn đề n...
Cách trị cảm cúm cho bà bầu bằng phương pháp dân gian	Khi mang thai chị em cần nằm lòng những cách trị cảm...
Ăn lựu bị vô sinh: Dựa vào đâu nói thế?	Ai cũng biết rằng, lựu là một trong những loại quả giàu...
Hồng ban nút - hậu quả của nhiều bệnh lý	Hồng ban nút là tình trạng viêm các tế bào mỡ dưới da...
Xử lý chứng hiếu động quá mức ở trẻ	Các nhà tâm lý đã nhìn nhận, trẻ mắc hội chứng thiếu t...
5 Bác sĩ đầu ngành về chuyên khoa tim mạch ở Hồ Chí...	Theo thống kê của Tổ chức Y tế thế giới (WHO), tỉ lệ tử...
Những tuyến xe bus đi qua Bệnh viện Phụ sản Trung ươ...	Trong các loại phương tiện giao thông, xe bus được coi...

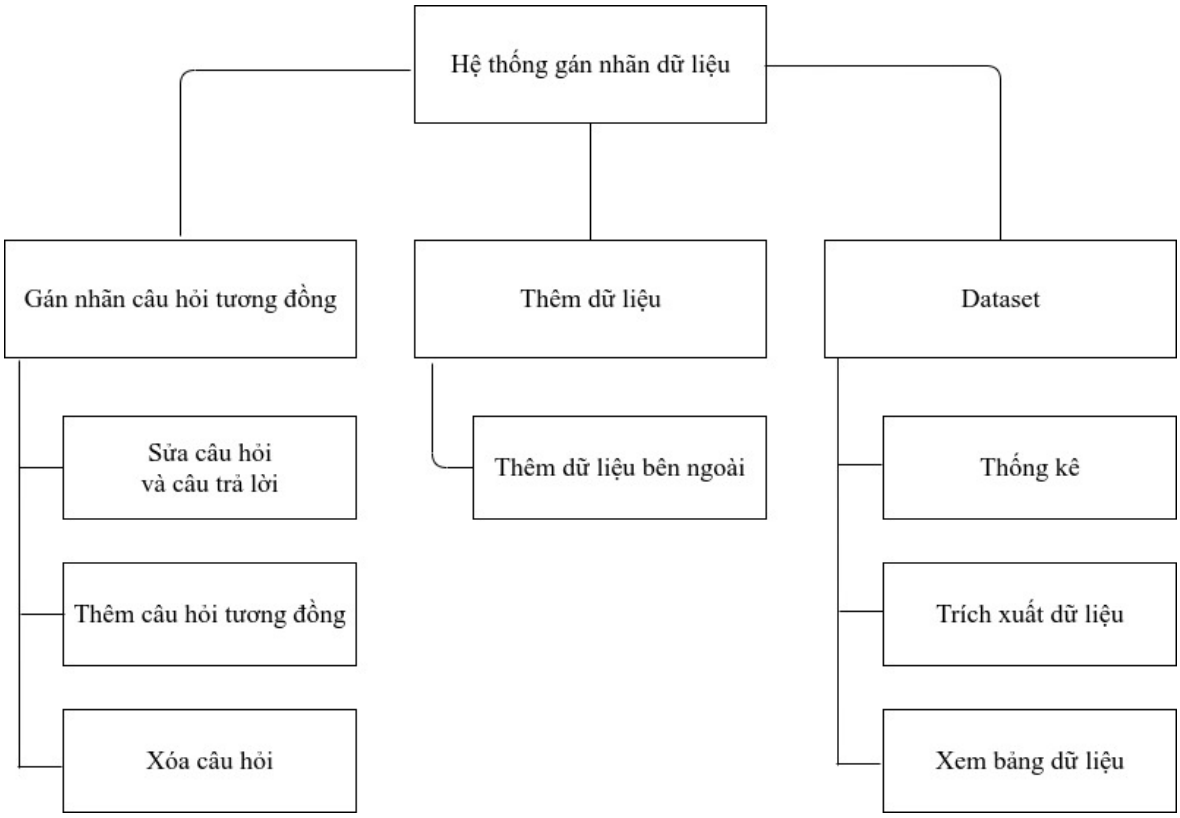
Hình 3.1: Dữ liệu thu thập được sau khi crawl

3.1.2 Xây dựng hệ thống gán nhãn dữ liệu

Sau khi đã có bộ dữ liệu là các cặp câu hỏi và câu trả lời trong lĩnh vực y tế. Công việc tiếp theo là cần sinh ra các cặp câu hỏi tương đồng và không tương đồng nhằm phục vụ cho mục đích huấn luyện và thực nghiệm các mô hình sau này. Với các câu hỏi không tương đồng, chúng ta có thể ghép ngẫu nhiên các câu hỏi trong bộ dữ liệu với nhau, sau đó người gán nhãn sẽ kiểm tra và xác nhận lại sự không tương đồng của các cặp câu hỏi đó. Với câu hỏi tương đồng, chúng ta phải xây dựng một hệ thống gán nhãn dữ liệu, việc gán nhãn được thực hiện bởi các bác sĩ, sinh viên chuyên ngành Y. Hệ thống phải đáp ứng được các yêu cầu như sửa lại câu hỏi và câu trả lời nếu chưa chính xác, thêm câu hỏi tương đồng cho câu hỏi hiện tại. Để xây dựng nhanh và hiệu quả một trang web hỗ trợ việc gán nhãn dữ liệu, bước này sử dụng thư viện **Streamlit** trên ngôn ngữ Python.

Streamlit là công cụ hữu ích cho các kỹ sư Machine learning, là một thư viện python mã nguồn mở có khả năng biến các tập lệnh dữ liệu thành các ứng dụng web có thể chia sẻ trong vòng vài phút. Nó hỗ trợ bằng các widget có sẵn, giúp chúng ta có thể xây dựng một web với tối thiểu các dòng lệnh cũng như các kiến thức về front-end. Ngoài ra, streamlit tích hợp các công cụ giúp trực quan hóa dữ liệu và tương tác người

dùng rất mạnh, chúng ta có thể tích hợp các thư viện để mô phỏng dữ liệu như pandas, matplotlib,... một cách dễ dàng. Về cơ bản, chúng ta có rất nhiều lựa chọn khác cho việc xây dựng một hệ thống gán nhãn cùng dựa trên ngôn ngữ Python như Django, Flask, ... Tuy nhiên, hệ thống gán nhãn mang tính chất nội bộ và cần xây dựng nhanh. Vì vậy, streamlit là một lựa chọn hợp lý và hiệu quả cho bước này.



Hình 3.2: Biểu đồ phân cấp chức năng cho hệ thống gán nhãn dữ liệu

Hình 3.2 mô tả biểu đồ phân cấp chức năng của hệ thống gán nhãn dữ liệu. Chức năng chính của hệ thống gồm 3 thành phần: Gán nhãn câu hỏi tương đồng cho bộ dữ liệu câu hỏi, câu trả lời đã crawl về; thêm dữ liệu từ nguồn bên ngoài; Dataset để thống kê dữ liệu, trích xuất và xem bảng dữ liệu tổng quan hiện tại.

Gán nhãn câu hỏi tương đồng

Trang gán nhãn câu hỏi tương đồng là trang quan trọng nhất của hệ thống. Tại đây, các chuyên gia, bác sĩ có thể gán nhãn các câu hỏi tương đồng với câu hỏi - câu trả lời tương ứng đã crawl về của bộ dữ liệu. Chúng ta có thể sửa câu hỏi và câu trả lời nếu như chúng chưa phù hợp, thêm các câu hỏi tương đồng với câu hỏi hiện tại.

Ngoài ra thì trang gán nhãn câu hỏi tương đồng có các chức năng như xóa câu hỏi, xóa nhãn đã gán cho các dữ liệu rác ngoài ý muốn.

Hình 3.3: Trang gán nhãn câu hỏi tương đồng

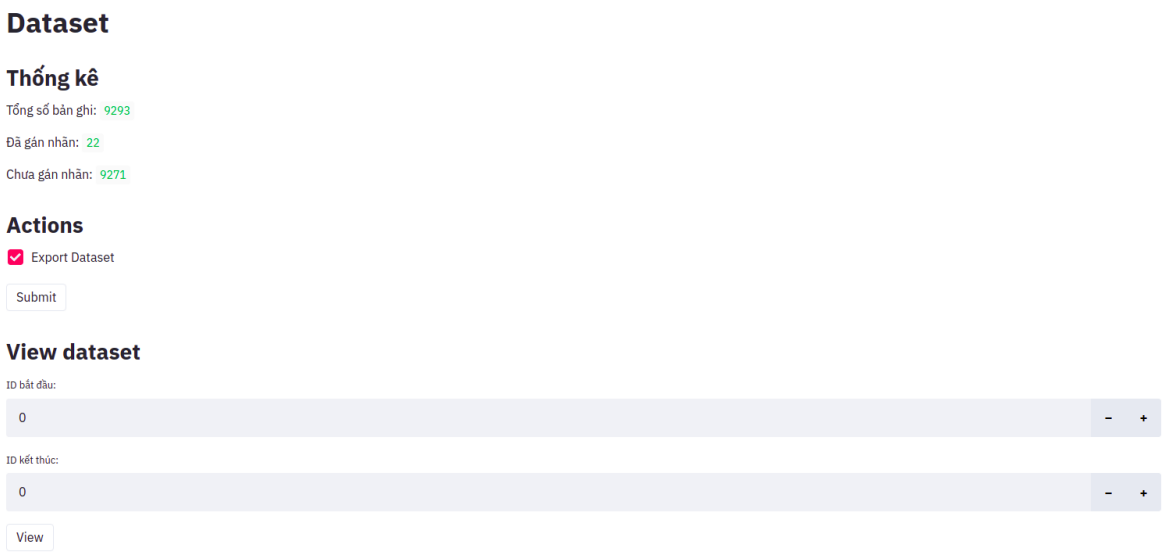
Thêm dữ liệu

Chức năng chính của trang thêm dữ liệu này là thêm câu hỏi, câu trả lời vào. Các chuyên gia có thể tùy ý thêm luôn các câu hỏi tương đồng cho dữ liệu đó hay không. Nếu có thì dữ liệu đó sẽ được coi như là đã gán nhãn trong tập dữ liệu. Ngược lại, nếu không có câu hỏi tương đồng cho dữ liệu mới thêm vào, thì dữ liệu đó sẽ được coi như là chưa gán nhãn và xuất hiện lại ở trang gán nhãn câu hỏi tương đồng để nhắc nhở người dùng gán nhãn.

Hình 3.4: Trang thêm dữ liệu

Dataset

Dataset là trang thống kê liên quan đến dữ liệu. Tại đây, người dùng có thể xem thống kê đã có bao nhiêu dữ liệu đã được gán nhãn và chưa gán nhãn (Phần Thống kê), trích xuất dữ liệu (Phần Actions) và xem bảng dữ liệu hiện tại (Phần View dataset). Hình 3.5 là giao diện của trang Dataset.



Hình 3.5: Trang Dataset

Để trích xuất dữ liệu về máy tính cá nhân, chúng ta tích vào *Export Dataset* và ấn *Submit*. Một liên kết để tải về dữ liệu sẽ hiện ra cho phép người dùng tải về. Để xem bảng dữ liệu hiện tại, nhập *ID bắt đầu* và *ID kết thúc* của dữ liệu mà muốn xem. Sau đó ấn *View* để hiện tình trạng bảng dữ liệu trong khoảng ID đó (Hình 3.6).

	answer	question	is_labeled	question_similaries
1	Triệu chứng lâm sàng của u gan thường nghèo nàn và không đặc hiệu: – Đau tức hạ sườn phải do khối u gây căng giãn bao glisson hoặc do chèn ép. – Ăn kém, gầy sút cân. – Hiếm khi có các triệu chứng tắc mật do khối u chèn ép vào đường mật. – Hiếm gặp một số trường hợp bệnh nhân đến viện vì biến chứng của u như u vỡ gây chảy máu trong ổ bụng, hoại tử u gây sốt và đau nhiều hoặc có khi u áp xe gây viêm phúc mạc.	Triệu chứng lâm sàng của bệnh u máu gan là gì thưa bác sĩ?	1	['Các triệu chứng của bệnh u máu gan?', 'Dấu hiệu của u máu gan?']
2	U máu gan là dị dạng mạch máu bẩm sinh và là khối u lành tính hay gặp nhất ở gan, với tỷ lệ 0,4 – 20% trên siêu âm và trên giải phẫu thực nghiệm. Chúng có nguồn gốc bẩm sinh và hiếm thấy có ung thư hóa.	Thưa bác sĩ, bệnh u máu gan có hay gặp không ạ?	0	[]
3	– Ung thư túi mật gặp nhiều ở nữ, tuổi cao, tỷ lệ nữ : nam là 9 : 4 và tuổi trung bình gặp trên 60 tuổi. – Ung thư túi mật liên quan chặt chẽ đến bệnh lý sỏi túi mật, sỏi túi mật chính là một tác nhân kích thích cơ học mãn tính từ đó dẫn đến tổn thương thoái hóa ác tính, hơn nữa người ta thấy có sự trùng hợp giữa vị trí phát sinh ung thư và vị trí của viên sỏi. – Ngoài ra một số tổn thương như: Polyp túi mật, túi mật teo, hóa sứ được xem như tổn thương tiền ung thư túi mật.	Ai có nguy cơ cao mắc bệnh ung thư túi mật ạ?	0	[]
4	Trong hầu hết các trường hợp trước mổ do đau, chân ít vận động trong thời gian dài làm cho cơ đùi, mông teo nhỏ. Vì vậy sau mổ, đặc biệt trong 8 tuần đầu, nguy cơ trật khớp háng nhân tạo tăng cao. Thời gian sau đó, nếu người bệnh tập luyện tốt, cơ rắn chắc trở lại thì nguy cơ này sẽ giảm dần. Trong quá trình tập luyện, người bệnh có thể bước đi trên khớp háng mới thay, tuy nhiên mới đầu sẽ đau và có cảm giác cứng khớp. Khi đó bác sĩ phẫu thuật sẽ khuyên bệnh nhân nên dùng nạng hỗ trợ khi đi lại và tỳ chân mức độ hợp lý, cùng với quy trình tập luyện được bác sĩ phục hồi chức năng đưa ra,	Sau thay khớp háng nhân tạo, người bệnh cần chú ý gì ạ?	0	[]

Hình 3.6: Trích xuất và hiển thị bảng dữ liệu trên trang Dataset

Sau khi đã xây dựng xong các chức năng cần thiết cho hệ thống gán nhãn dữ liệu. Chúng ta cần deploy hệ thống lên server. Dưới đây là các thông số cấu hình để deploy sản phẩm.

config.toml

```
[server]
headless = true
port = 5000
enableCORS = false
```

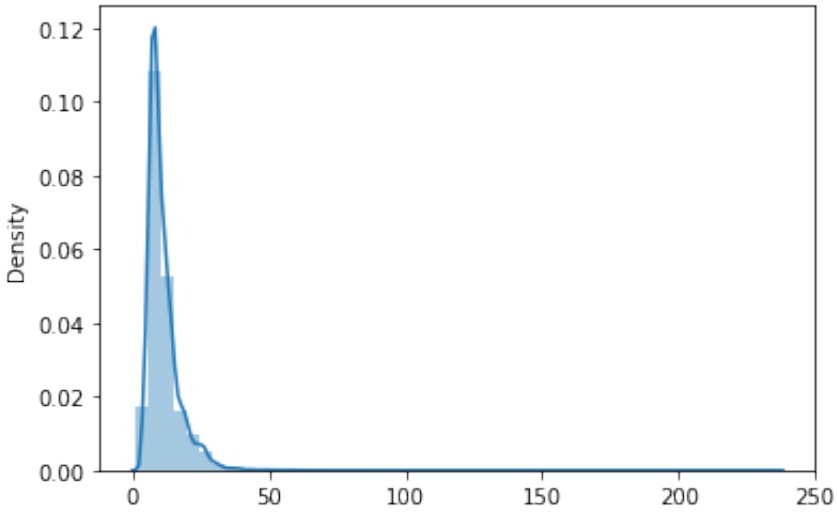
dockerfile

```
FROM python:3.7
WORKDIR /app
COPY requirements.txt /app
RUN pip install -r requirements.txt
COPY . /app
RUN mkdir -p .streamlit
RUN mv config.toml .streamlit
CMD ["streamlit", "run", "app.py"]
```

requirements.txt

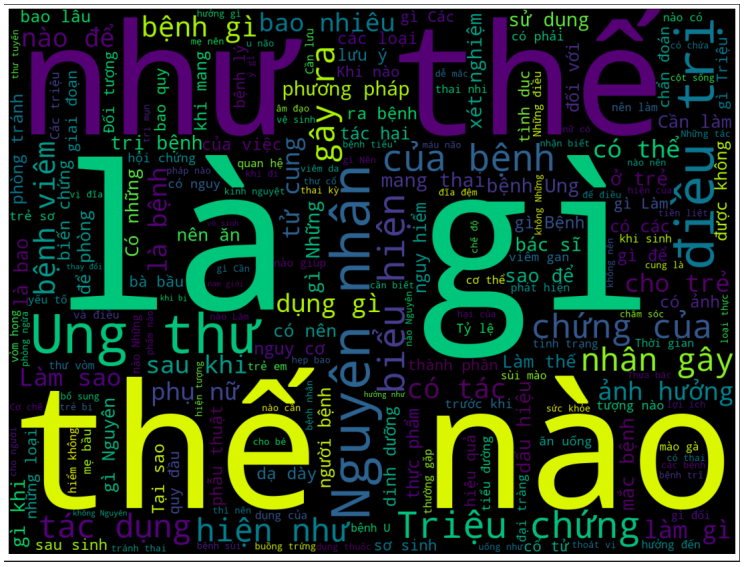
```
pandas
numpy
streamlit
xlsxwriter
matplotlib
gensim==3.8.3
```

3.2 Phân tích dữ liệu



Hình 3.7: Đồ thị biểu diễn phân phối số từ của các câu hỏi

Từ biểu đồ 3.7, có thể thấy rằng, các câu hỏi có số từ trung bình thường dưới 50 từ. Vì vậy, khi thiết lập mô hình, chúng ta có thể thiết lập max length cho đầu vào của mô hình là 50.



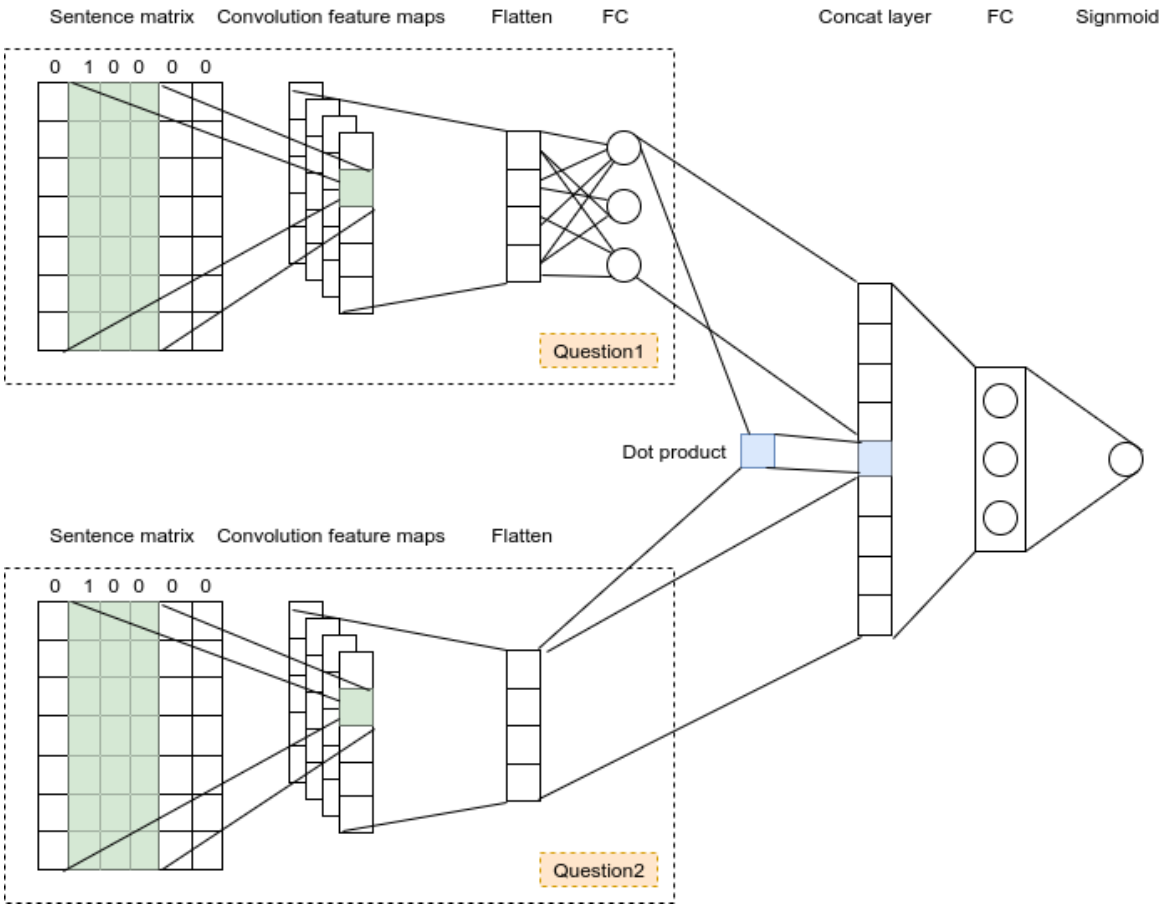
Hình 3.8: Các từ xuất hiện nhiều nhất trong dữ liệu

Một số từ xuất hiện nhiều nhất trong dữ liệu thường là stop word hoặc các từ mang ý nghĩa để hỏi. Dữ liệu negative được tạo để cân bằng với positive, vì vậy số thực thể của 2 class là cân bằng nhau. Do đó để đánh giá mô hình, chúng ta có thể dùng độ đo accuracy.

3.3 Kết quả thực nghiệm các mô hình

3.3.1 Mô hình CNN

Trong thực nghiệm này, em áp dụng mô hình CNN cho xử lý ngôn ngữ tự nhiên. Các khối chính là hai mô hình phân phối của câu dựa trên mạng CNN. Chúng hoạt động song song, ánh xạ hai câu hỏi thành các vector phân phối, sau đó các vector này được sử dụng để học các thông tin ngữ nghĩa tương đồng giữa chúng. Mô hình thực nghiệm mạng CNN cho bài toán được mô tả bởi hình 3.9. Thông tin chi tiết các layer và tham số được mô tả ở bảng 3.1.



Hình 3.9: Mô hình thực nghiệm mạng CNN cho bài toán

Layer (type)	Ouput Shape	Param	Connect to
input_1 (InputLayer)	[(None, 50)]	0	
input_2 (InputLayer)	[(None, 50)]	0	
embedding (Embeeding)	(None, 50, 300)	1169100	input_1[0][0] input_2[0][0]
conv1d (Conv1D)	(None, 48, 200)	180200	embedding[0][0] embedding[1][0]
max_pooling1d (MaxPooling1D)	(None, 24, 200)	0	conv1d[0][0]
flatten (Flatten)	(None, 4800)	0	max_pooling1d[0][0]
max_pooling1d_1 (MaxPooling1D)	(None, 24, 200)	0	conv1d[1][0]
flatten_1 (Flatten)	(None, 4800)	0	max_pooling1d_1[0][0]
dense (Dense)	(None, 4800)	23044800	flatten[0][0]
dot (Dot)	(None, 1)	1	dense[0][0] flatten_1[0][0]
tf.concat (TfOpLambda)	(None, 9601)	0	flatten[0][0] dot[0][0] flatten_1[0][0]
dropout (Dropout)	(None, 9601)	0	tf.concat[0][0]
dense_1 (Dense)	(None, 100)	960200	dropout[0][0]
dense_2 (Dense)	(None, 1)	101	dense_1[0][0]

Bảng 3.1: Kiến trúc mô hình CNN thực nghiệm

Sentence matrix:

Đầu vào của mạng là các câu với các từ cần chuyển đổi sang các vector giá trị thực để có thể xử lý qua các lớp trong mạng. Trong phần này, dữ liệu được tokenizer bằng thư viện **pyvi**. Ví dụ với câu:

Trường đại học bách khoa hà nội

sẽ được tokenizer thành:

Trường đại_học bách_khoa hà_nội

Đầu vào của mạng là câu có độ dài cố định là s biểu diễn bởi các từ: $[w_!, \ldots, w_s]$ trong bộ từ điển hữu hạn $|V|$. Với những câu đầu vào không đủ độ dài s, chúng ta sẽ thêm các từ $< PAD >$ đại diện cho viền của câu đó vào cuối cho đến khi đủ độ dài s. Với những câu vượt quá độ dài s, thực hiện việc cắt các từ ở cuối câu đi để chuẩn hóa độ dài của câu về s từ. Các từ w_i này được chuyển sang số tương ứng với số thứ tự của nó trong từ điển. Ví dụ, ta có bộ từ điển như sau:

{'Trường':0, 'đại_học':1, 'bách_khoa': 2, 'hà_nội':3}

thì câu đầu vào ở ví dụ trên sẽ được biểu diễn thành:

[[0], [1], [2], [3]]

Kiến trúc mạng neural không phù hợp để xử lý các từ rời rạc, vì thế, chúng ta biểu diễn các từ đầu vào với số chiều thấp hơn và giá trị thực $w \in \mathbb{R}^{d_w}$ được look up từ ma trận $W \in \mathbb{R}^{d_w \times |V|}$ (Các cột ma trận này tương ứng với các từ trong $|V|$). Chúng ta ánh xạ các từ qua ma trận và thu được word embedding của nó bằng toán tử look up table $LT_W(w_i) = \omega_i$. Do đó, với mỗi câu đầu vào độ dài s, chúng ta xây dựng một ma trận S với cột thứ i là word embeding của từ w_i. Ma trận này gọi là **Sentence matrix**. Để học các đặc trưng ngữ nghĩa của câu, mạng sử dụng lớp convolution và pooling để trích xuất các đặc trưng qua ma trận S.

Convolution feature maps:

Mục đích của lớp convolution là trích xuất các mẫu. Với đầu vào là ma trận $S \in \mathbb{R}^{d \times |s|}$ và filter (hoặc kernel) $F \in \mathbb{R}^{d \times m}$, thực hiện toán tử convolution $*$ giữa chúng, thu được vector $c \in \mathbb{R}^{|s|+m-1}$ với các thành phần cho bởi:

$$c_i = (S * F)_i = \sum_{k,j} (S_{[:,i-m+1:i]} \otimes F)_{kj}$$

trong đó $\otimes$ là toán tử nhân element-wise và $S_{[:,i-m+1:i]}$ là ma trận slide kích cỡ m dọc theo các cột.

Để tăng tính biểu diễn dữ liệu, mô hình học sâu áp dụng nhiều filter chạy song song để sinh các feature map (Như hình 3.9). Khi sử dụng một tập các filter $F \in \mathbb{R}^{n \times d \times m}$ trên ma trận S thu được một ma trận feature map $C \in \mathbb{R}^{n \times (|s|-m+1)}$, ma trận này được flatten ra thành một vector. Trong mô hình trên, vector cho câu hỏi thứ nhất được tiếp tục đi qua một lớp kết nối dày đặc thu được x_{q_1} và thực hiện dot product với vector biểu diễn cho câu hỏi thứ hai x_{q_2}. Hàm kích hoạt được sử dụng mỗi lớp convolution là hàm $ReLU(x) = \max(0, x)$.

Dot layer:

Lớp Dot thực hiện toán tử dot product giữa hai vector. Trong mô hình, chúng ta thực hiện dot giữa hai vector biểu diễn cho hai câu hỏi và thu được một vector $x_{dot} \in \mathbb{R}$.

Concat và hidden layer:

Cuối cùng, mô hình thực hiện nối các vector đặc trưng lại với nhau để thành đầu vào cho một mạng phân loại. Vector biểu diễn câu hỏi thứ nhất và giá trị dot product và vector biểu diễn câu hỏi thứ hai được nối lại thành 1 vector:

$$x_{joint} = [x_{q_1}; x_{dot}; x_{q_2}]$$

và đưa qua một mạng kết nối đầy đủ để thực hiện phân loại. Đầu ra của mạng là 1 unit sử dụng hàm sigmoid để đưa các giá trị kết quả về khoảng $[0, 1]$.

Training:

Hàm loss function được sử dụng là hàm binary cross-entropy:

$$L = -\hat{y}\log(\hat{y}) - (1 - \hat{y})\log(1 - \hat{y})$$

trong đó $\hat{y}$ là đầu ra của mô hình. Và phương pháp tối ưu được sử dụng là phương pháp Adam.

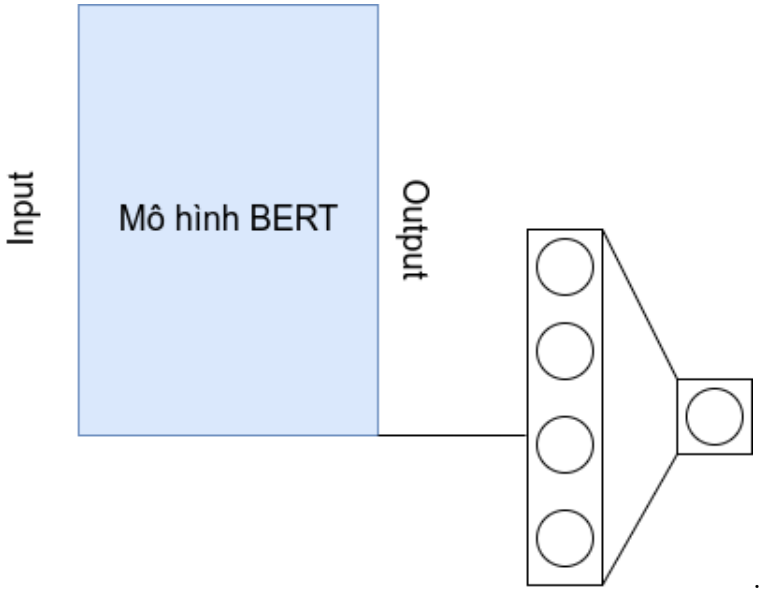
Kết quả

Thực hiện training 50 epoch, độ chính xác đạt 81.17%.

3.3.2 Mô hình PhoBert

Kiến trúc mô hình

Em thực nghiệm mô hình Phobert như hình 3.10. Thư viện transformer được sử dụng trong thực nghiệm này, đầu ra của lớp cuối cùng của mô hình được đưa vào một mạng kết nối đầy đủ để thực hiện phân loại.



Hình 3.10: Mô hình PhoBert thực nghiệm

Kiến trúc Roberta Tokenizer sử dụng byte-level Byte-Pair-Encoding. Dữ liệu đưa vào mô hình gồm 2 câu hỏi được tokenization thành các sub-word và nối với nhau, các token <s> và </s> được sử dụng để phân biệt giữa 2 câu hỏi, token <pad> sử dụng để đánh dấu là phần padding.

Kết quả

Thực hiện training 100 epoch với learning rate là 3e-5. Độ chính xác đạt 72.8%

3.3.3 Mô hình LightGBM

Trích chọn đặc trưng

Việc lựa chọn các đặc trưng cho mô hình là một trong những phần quan trọng quyết định đến kết quả của mô hình Ensemble. Chọn đặc trưng không tốt có thể dẫn tới hiệu suất của mô hình không cao, và ngược lại, nếu chọn được các đặc trưng có ý nghĩa cao sẽ giúp mô hình đạt kết quả tốt trên bài toán đặt ra. Với kinh nghiệm của bản thân và tham khảo từ các bài báo khoa học, em lựa chọn các đặc trưng sau cho mô hình:

- **word match:** Tỉ lệ của trọng số các từ chung của 2 câu hỏi và tổng trọng số của các từ trong 2 câu hỏi. Trong đó trọng số của một từ w được định nghĩa là 0 nếu nó xuất hiện ít hơn 2 lần trong bộ dữ liệu, ngược lại được định nghĩa là $weight(w) = 1/(count + eps)$, với count là số lần xuất hiện của từ w trong bộ dữ liệu, eps là một hằng số.
- **word match 2 root:** Căn bậc 2 của word match.
- **TFIDF word match:** Tỉ lệ của số lượng các từ chung của 2 câu hỏi và tổng số từ của 2 câu (tổng số từ chỉ đếm mỗi từ 1 lần).
- **share count:** Số từ chung của 2 câu hỏi.
- **share 2gram:** Tỉ lệ số cặp 2 gram chung trên tổng số cặp 2 gram của 2 câu hỏi.
- **consine:** Khoảng cách consine giữa 2 weight của 2 câu hỏi.
- **words hamming:** Số các ký tự ở vị trí tương đương có giá trị giống nhau chia cho độ dài của câu lớn hơn.
- **len question 1:** Số ký tự của câu hỏi thứ nhất, bao gồm cả khoảng trắng.
- **len question 2:** Số ký tự của câu hỏi thứ 2, bao gồm cả khoảng trắng.
- **different length:** Chênh lệch giữa số ký tự câu hỏi thứ nhất và thứ 2.
- **caps count question 1:** số ký tự viết hoa trong câu hỏi thứ nhất.
- **caps count question 2:** Số ký tự viết hoa trong câu hỏi thứ hai.
- **diferrent caps:** Chênh lệch giữa số ký tự viết hoa của câu hỏi thứ nhất và câu hỏi thứ hai.
- **length character question 1:** Số ký tự của câu hỏi thứ nhất, không bao gồm khoảng trắng.
- **length character question 2:** Số ký tự của câu hỏi thứ hai, không bao gồm khoảng trắng.

- **differrent length character:** Chênh lệch giữa số ký tự câu hỏi thứ nhất và thứ hai, không bao gồm khoảng trắng.
- **length word question 1:** Số từ của câu hỏi thứ nhất.
- **length word question 2:** Số từ của câu hỏi thứ hai.
- **differrent length word:** Chênh lệch giữa số từ câu hỏi thứ nhất và thứ hai.
- **extractly same:** Nhận giá trị 1 nếu hai câu hỏi giống nhau, ngược lại trả về 0.

Cấu hình thực nghiệm

LightGBM được thực nghiệm sử dụng thư viện có sẵn của python là **lightgbm**. Dưới đây là các tham số cấu hình cho mô hình này:

- Cấu hình thực nghiệm cho LightGBM:

```
learning_rate: 0.01
boosting_type: 'dart'
application: 'binary'
objective: 'binary'
metric: 'auc'
sub_feature: 0.5
num_leaves: 31
min_data: 50
min_hessian: 1
verbose: 1
```

Kết quả

Accuracy: 92.26%

KẾT LUẬN

Như vậy, qua đồ án này, em đã thực hiện xây dựng dữ liệu bộ câu hỏi chung về lĩnh vực y tế và thực nghiệm các mô hình cho hệ thống hỏi đáp trong lĩnh vực y tế dựa trên sự tương đồng câu hỏi. Qua đồ án này, em đã thực hiện được những việc sau:

- Crawl dữ liệu câu hỏi - câu trả lời tiếng Việt cho lĩnh vực y tế.
- Gán nhãn các câu hỏi tương đồng.
- Thực nghiệm các mô hình.

Do vì thời gian có hạn, quá trình xây dựng đồ án mất nhiều thời gian vào việc xây dựng bộ dữ liệu. Vì vậy, đồ án còn nhiều thiết sót, mong được sự góp ý của thầy cô và các bạn. Đây là một số hướng phát triển tương lai cho đồ án này:

- Cải tiến các mô hình thực nghiệm, xây dựng bổ sung các bộ dữ liệu chuyên sâu vào từng mảng trong lĩnh vực y tế.
- Xây dựng mô hình trích xuất các đặc trưng từ câu hỏi như: triệu chứng, tên bệnh, ... Biểu diễn các đặc trưng này thành dạng dữ liệu đồ thị để có thể cải tiến độ chính xác và đưa ra phản hồi hợp lý cho người dùng.

Tài liệu tham khảo

[1] McCreery, C., Katariya, N., Kannan, A., Chablani, M. and Amatriain, X., 2019. "Domain-Relevant Embeddings for Medical Question Similarity". arXiv preprint arXiv:1910.04192.

[2] L. Sharma, L. Graesser, N. Nangia, and U. Evci, "Natural language understanding with the quora question pairs dataset," CoRR, vol. abs/1907.01041, 2019.

[3] A. Severyn and A. Moschitti, "Modeling relational information in question-answer pairs with convolutional neural networks," arXiv preprint arXiv:1604.01178, 2016.

[4] Martinez, D., MacKinlay, A., Aliod, D. M., Cavedon, L., & Verspoor, K. "Simple Similarity-based Question Answering Strategies for Biomedical Text". In CLEF.

[5] Q. Bao, L. Ni, and J. Liu, "HHH: an online medical chatbot system based on knowledge graph and hierarchical bi-directional attention," Proceedings of the Australasian Computer Science Week Multiconference, vol. 7, 2020.

[6] Yinfei Yang, Steve Yuan, Daniel Cer, Sheng yi Kong, Noah Constant, Petr Pilar, Heming Ge, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. "Learning semantic textual similarity from conversations." Proceedings of RepL4NLP workshop at ACL.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need." arXiv preprint arXiv:1706.03762, 2017.

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805, 2018.

[9] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. "Roberta: A robustly optimized bert pretraining approach." arXiv preprint arXiv:1907.11692, 2019.

[10] Darsh J Shah, Tao Lei, Alessandro Moschitti, Salvatore Romeo, and Preslav Nakov. 2018. "Adversarial domain adaptation for duplicate question detection." In EMNLP.

[11] Nguyen, D. Q. and Nguyen, A. T. (2020). Phobert: Pretrained language models for vietnamese. arXiv preprint arXiv:2003.00744.

[12] T. Chen and C. Guestrin. "Xgboost: A scalable tree boosting system." In Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 785–794. ACM, 2016.

[13] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. "Lightgbm: A highly efficient gradient boosting decision tree." In Advances in Neural Information Processing Systems, pages 3149–3157, 2017.