ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA TOÁN - CƠ - TIN HỌC

BÁO CÁO TIỂU LUẬN CUỐI KỲ

Applying PhoBERT Encoder for Sentiment Classification

Ngành: Khoa học dữ liệu (Chương trình đào tạo chuẩn)

Sinh viên thực hiện: Hán Minh Thành – 24001699
Giảng viên hướng dẫn: TS. Hoàng Anh Đức

Hà Nội - 2025

Thông tin Dự án

Học phần: MAT3508 – Nhập môn Trí tuệ Nhân tạo

Học kỳ: Học kỳ 1, Năm học 2025-2026

Trường: VNU-HUS (Đại học Quốc gia Hà Nội – Trường Đại học Khoa học Tự nhiên)

Tên dự án: Applying PhoBERT Encoder for Sentiment Classification

Ngày nộp: 30/11/2025

Thành viên nhóm

Họ tên	**Mã sinh viên**	**Tên GitHub**	**Đóng góp**
Hán Minh Thành	24001699	24001699-lgtm	Thực hiện dự án

Giảng viên hướng dẫn: TS. Hoàng Anh Đức

Lời cảm ơn

Đầu tiên, em xin gửi lời cảm ơn chân thành và sâu sắc nhất đến **TS. Hoàng Anh Đức**, người đã tận tình hướng dẫn và giúp đỡ em trong suốt quá trình thực hiện tiểu luận. Em xin trân trọng cảm ơn các Thầy, Cô trong **Khoa Toán - Cơ - Tin học** đã tận tâm truyền đạt kiến thức và tạo môi trường nghiên cứu thuận lợi. Mặc dù đã nỗ lực cố gắng, nhưng do kiến thức và kinh nghiệm còn hạn chế nên bài tiểu luận khó tránh khỏi những thiếu sót. Em rất mong nhận được sự thông cảm và những ý kiến đóng góp quý báu của Thầy để bài làm được hoàn thiện hơn.

Em xin chân thành cảm ơn!

Sinh viên thực hiện:
Hán Minh Thành

Tóm tắt nội dung

Báo cáo này trình bày quá trình nghiên cứu và ứng dụng mô hình **PhoBERT** cho bài toán phân loại cảm xúc tiếng Việt, sử dụng dữ liệu phản hồi của khách hàng trong lĩnh vực dược phẩm. Nội dung báo cáo bao gồm các giai đoạn: thu thập dữ liệu (*crawl*), tiền xử lý, gán nhãn bằng *Doccano*, gán nhãn thực thể (NER), tinh chỉnh (*fine-tune*) mô hình PhoBERT . Kết quả thực nghiệm cho thấy PhoBERT đạt độ chính xác cao, hoạt động ổn định và có tiềm năng ứng dụng thực tế trong các hệ thống phân tích cảm xúc tiếng Việt.

Mục lục

Lời cảm ơn **2**

Tóm tắt nội dung (Abstract) **3**

1 Khảo sát bài toán và yêu cầu hệ thống phân tích cảm xúc tiếng Việt **5**
1.1 Mục đích chọn đề tài 5
1.2 Khảo sát bài toán 5
1.2.1 Khảo sát yêu cầu thực tế của bài toán 5
1.2.2 Chức năng hệ thống 6

2 Phân tích và xử lí dữ liệu **7**
2.1 Tiền xử lý dữ liệu 7
2.2 Gán nhãn dữ liệu 7
2.3 Gán nhãn thực thể (Named Entity Recognition - NER) 9

3 Mô hình xử lý ngôn ngữ tự nhiên - PhoBERT **10**
3.1 Giới thiệu 10
3.2 Kiến trúc của BERT 11
3.3 Fine-tuning cho tác vụ Classification 12

4 Tổng kết và Hướng phát triển **17**
4.1 Tổng kết quá trình thực hiện 17
4.2 Hướng phát triển 17

1. Khảo sát bài toán và yêu cầu hệ thống phân tích cảm xúc tiếng Việt

1.1 Mục đích chọn đề tài

Trong kỷ nguyên số, lượng dữ liệu phản hồi từ khách hàng trên các nền tảng trực tuyến tăng trưởng theo cấp số nhân. Việc phân tích thủ công không còn khả thi về mặt thời gian và chi phí. Do đó, nhu cầu ứng dụng Trí tuệ nhân tạo (AI) và Xử lý ngôn ngữ tự nhiên (NLP) để tự động hóa quy trình này trở nên cấp thiết hơn bao giờ hết. Mục tiêu của đề tài là xây dựng hệ thống tự động nhận diện sắc thái cảm xúc, giúp doanh nghiệp chuyển đổi dữ liệu thô thành thông tin chi tiết có giá trị, từ đó nâng cao lợi thế cạnh tranh...

Qua quá trình nghiên cứu, em đã tiến hành phân tích và xử lý một lượng lớn dữ liệu thu thập từ các phản hồi của khách hàng, sử dụng các phương pháp máy học tiên tiến để phân loại các cảm xúc và phản ứng của họ. Dự án không chỉ mang lại cơ hội để áp dụng các kiến thức đã học vào thực tiễn, mà còn giúp em hiểu rõ hơn về tầm quan trọng của việc lắng nghe và phân tích những phản hồi từ khách hàng trong việc định hình các chiến lược kinh doanh hiệu quả, qua đó rút được ra kinh nghiệm áp dụng cho thực tế.

1.2 Khảo sát bài toán

1.2.1 Khảo sát yêu cầu thực tế của bài toán

Khảo sát yêu cầu thực tế Nguồn dữ liệu là các phản hồi khách hàng trên các website thương mại điện tử và mạng xã hội hiện nay rất phong phú nhưng tồn tại dưới dạng phi cấu trúc và chứa nhiều nhiễu.Thách thức về đặc trưng ngôn ngữ: Khác với tiếng Anh, tiếng Việt là ngôn ngữ đơn âm tiết nhưng ngữ nghĩa lại phụ thuộc nhiều vào cách ghép từ và ngữ cảnh. Đặc biệt trên môi trường mạng xã hội, văn bản thường chứa nhiều teencode, viết tắt, sai chính tả, icon cảm xúc và cấu trúc ngữ pháp không chuẩn mực. Hệ thống cần có khả năng hiểu được các biến thể này để tránh phân loại sai lệch. Do đó, bài toán đặt ra yêu cầu cấp thiết về việc xây dựng cơ chế thu thập dữ liệu đa nguồn ổn định, đồng thời phải tuân thủ nghiêm ngặt các quy định về bảo mật và quyền riêng tư. Đặc biệt, hệ thống cần đạt độ chính xác cao trong khâu tiền xử lý và phân loại, đảm bảo tính khách quan khi áp dụng vào việc đánh giá hiệu suất làm việc của nhân sự trong thực tế.

1.2.2 Chức năng hệ thống

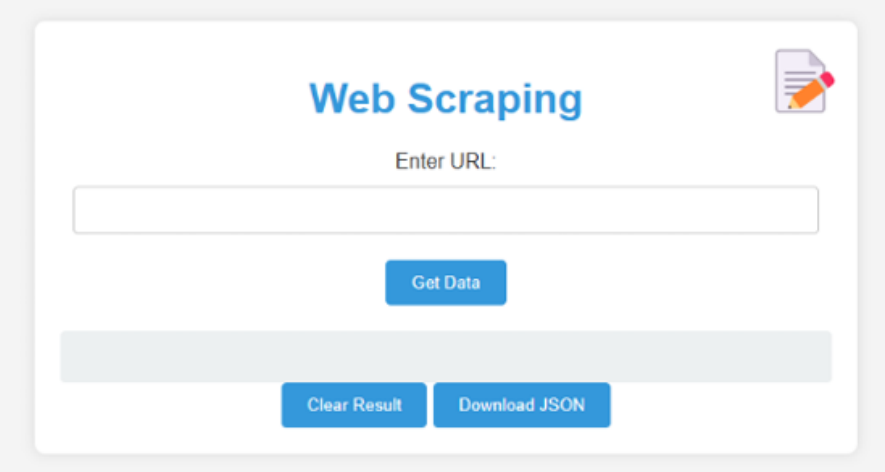
Để giải quyết các yêu cầu trên, hệ thống được thiết kế theo quy trình xử lý dữ liệu khép kín (pipeline) với các module chức năng cụ thể:

- **Crawler:** Tự động hóa việc thu thập dữ liệu thô từ các nguồn website và mạng xã hội đã xác định.
- **Tiền xử lý (Preprocessing):** Làm sạch dữ liệu, chuẩn hóa văn bản tiếng Việt và loại bỏ nhiễu để tối ưu hóa đầu vào.
- **Gán nhãn dữ liệu:** Sử dụng công cụ Doccano để gán nhãn cảm xúc và thực thể (NER), phục vụ việc xây dựng tập dữ liệu huấn luyện chất lượng cao.
- **Huấn luyện :** Xây dựng mô hình học máy để phân loại sắc thái phản hồi, đảm bảo các chỉ số đánh giá đạt mức tin cậy.
- **Lưu trữ và Báo cáo:** Quản lý kết quả phân tích và hiển thị dưới dạng Dashboard trực quan, hỗ trợ doanh nghiệp theo dõi và ra quyết định quản trị nhân sự kịp thời, đảm bảo chất lượng tốt nhất cho khách hàng tránh rủi ro không đáng có.

2. Phân tích và xử lí dữ liệu

2.1 Tiền xử lý dữ liệu

Trước khi tiến hành tiền xử lý, em đã sử dụng BeautifulSoup – một thư viện Python phổ biến để thu thập và trích xuất dữ liệu từ các trang web. Cụ thể, BeautifulSoup giúp em đọc và tách nội dung văn bản từ mã HTML, qua đó lấy được các đoạn văn, tiêu đề hoặc bình luận cần thiết cho tập dữ liệu. Trong quá trình này, em kết hợp với thư viện Requests để gửi yêu cầu đến trang web, sau đó dùng BeautifulSoup để loại bỏ các thẻ HTML, ký tự đặc biệt và các phần không cần thiết.



Hình 2.1: Sử dụng BeautifulSoup để crawl dữ liệu

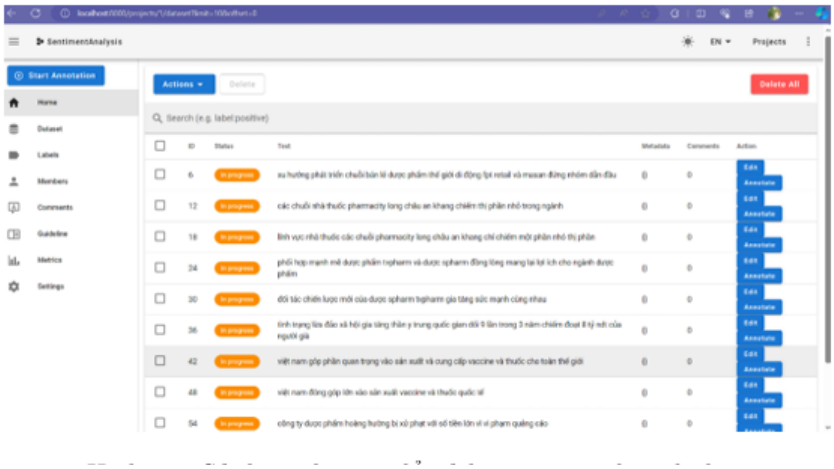
Sau khi thu thập, dữ liệu thô được đưa vào bước tiền xử lý để phục vụ cho quá trình phân tích và xây dựng mô hình. Em đã áp dụng công cụ VnTokenizer, giúp chia nhỏ văn bản thành các đơn vị tokens, làm sạch dữ liệu bằng cách loại bỏ ký tự không cần thiết và chuẩn hóa ngôn ngữ. Việc tiền xử lý này giúp nâng cao hiệu quả của quá trình huấn luyện và phân tích dữ liệu ở các bước tiếp theo.

2.2 Gán nhãn dữ liệu

Để chuẩn bị dữ liệu cho mô hình, em đã sử dụng nền tảng **Doccano** để gán nhãn cảm xúc cho các mẫu văn bản. Quá trình này bao gồm việc phân loại từng đoạn văn bản vào một trong ba nhãn cảm xúc chính: *NEG* (tiêu cực), *NEU* (trung tính), và *POS* (tích cực).

Hệ thống gán nhãn được thực hiện cẩn thận, có bước kiểm tra chéo nhằm đảm bảo tính chính xác và nhất quán của dữ liệu. Doccano cung cấp một giao diện trực quan, giúp

em dễ dàng xem, chọn và gán nhãn cảm xúc tương ứng với từng phản hồi của khách hàng hoặc từng câu văn.



Hình 2.2: Giao diện gán nhãn dữ liệu trong Doccano

Sau khi hoàn tất quá trình gán nhãn, Doccano cho phép xuất dữ liệu ra nhiều định dạng khác nhau như JSON, CSV hoặc JSONL. Trong báo cáo này, em sử dụng định dạng JSONL (JSON Lines), trong đó mỗi dòng tương ứng với một mẫu dữ liệu bao gồm hai thành phần: nội dung văn bản và nhãn cảm xúc tương ứng.

[Ví dụ dữ liệu xuất từ Doccano]

```
{"text": "Sản phẩm rất tốt, giao hàng nhanh.", "label": "2"}
{"text": "Chất lượng bình thường, không có gì đặc biệt.", "label": "1"}
{"text": "Dịch vụ quá tệ, nhân viên không nhiệt tình.", "label": "0"}
```

Trong đó:
- 2 – POS
- 1 – NEU
- 0 – NEG

Sau khi thu được bộ dữ liệu được gán nhãn đầy đủ, em nhận thấy vẫn tồn tại một vấn đề nhỏ ảnh hưởng đến chất lượng huấn luyện mô hình. Cụ thể, khi một câu chứa quá nhiều chủ thể hoặc thông tin không liên quan, mô hình có thể bị "nhiễu", khiến việc học đặc trưng cảm xúc trở nên kém hiệu quả. Trong bài toán phân loại cảm xúc, điều quan trọng nhất là nhận biết được những từ hoặc cụm từ mang ý nghĩa cảm xúc rõ rệt, thay vì toàn bộ nội dung.

Việc có quá nhiều chủ thể trong cùng một câu cũng dễ gây ra sự "nhầm lẫn" cho mô hình. Chẳng hạn, hai câu sau:

"Anh A mất vì bệnh K." và "Bệnh K đã lấy đi sinh mạng của anh A."

mặc dù mang cùng một ý nghĩa, nhưng khi được chuyển thành vector embedding, kết quả lại có thể khác nhau đáng kể. Điều này khiến mô hình có thể hiểu rằng hai câu thuộc hai lớp cảm xúc khác nhau, dù thực tế nội dung của chúng tương tự.

Để khắc phục vấn đề này, em bổ sung thêm một bước xử lý là gán nhãn thực thể (Entity Labeling). Bước này giúp mô hình nhận diện và xử lý tốt hơn các thực thể quan trọng trong câu (ví dụ: tên người, địa điểm, tổ chức...), đồng thời giảm ảnh hưởng của các từ ít liên quan, góp phần cải thiện độ chính xác của mô hình học máy.

2.3 Gán nhãn thực thể (Named Entity Recognition - NER)

Để có thể nhận diện được các thực thể trong câu và gán nhãn cho chúng, trước hết, chúng ta cần xác định một quy tắc gán nhãn thống nhất. Trong phạm vi báo cáo này, em chỉ sử dụng ba nhãn NER chính, bao gồm:

- **PER (Person – Người)**
  Mô tả: Nhãn này dùng để nhận diện các tên riêng của cá nhân, bao gồm tên đầy đủ, tên riêng hoặc bất kỳ tên gọi nào đề cập đến một cá nhân cụ thể.
  Ví dụ: `"Anh A vừa kết hôn với chị B"` → `PER vừa kết hôn với PER`

- **LOC (Location – Địa điểm)**
  Mô tả: Nhãn này dùng để nhận diện các địa danh, bao gồm tên quốc gia, thành phố, thị trấn, vùng, khu vực địa lý hoặc bất kỳ địa điểm cụ thể nào khác.
  Ví dụ: `"Anh A nhà ở Đồng Nai"` → `PER nhà ở LOC`

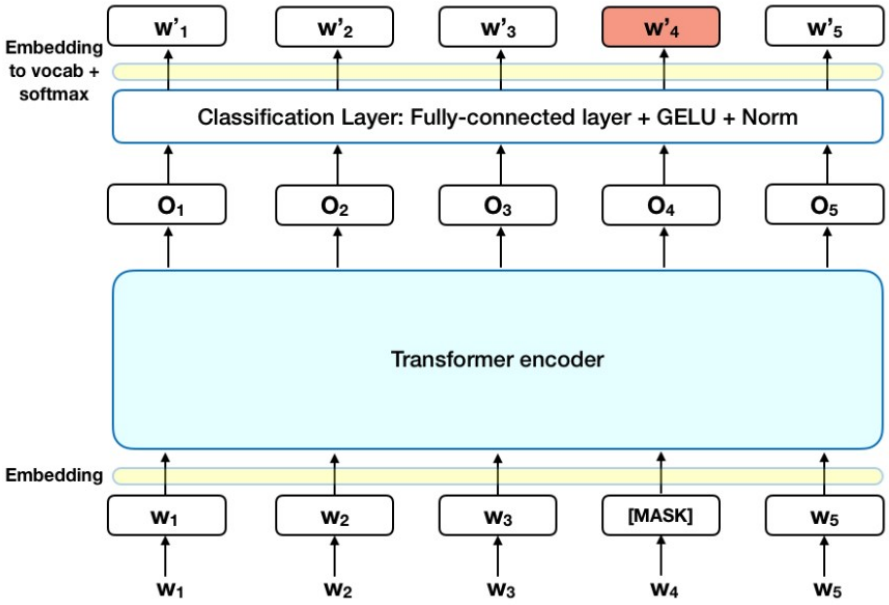
- **ORG (Organization – Tổ chức)**
  Mô tả: Nhãn này dùng để nhận diện các tên của tổ chức, bao gồm công ty, cơ quan chính phủ, tổ chức phi chính phủ, tổ chức quốc tế, trường học, bệnh viện, v.v.
  Ví dụ: `"Anh A học tại trường X"` → `PER học tại ORG`

Sau khi đã xác định quy tắc và danh sách các nhãn, tiến hành áp dụng mô hình để thực hiện việc gán nhãn thực thể. Mô hình được sử dụng trong báo cáo này là `NlpHUST/ner-vietnamese-electra-base` được cung cấp trên nền tảng **Hugging Face**. Mô hình này được huấn luyện trên tập dữ liệu tiếng Việt và có khả năng nhận diện hiệu quả các thực thể phổ biến trong văn bản tiếng Việt.

3. Mô hình xử lý ngôn ngữ tự nhiên - PhoBERT

3.1 Giới thiệu

Mô hình cuối cùng được nhắc đến và sử dụng trong báo cáo lần này là mô hình PhoBERT. PhoBERT là một mô hình *pre-trained* được huấn luyện trên ngôn ngữ đơn (monolingual language) là tiếng Việt. Nó dựa trên kiến trúc và cách tiếp cận giống RoBERTa (Liu et al., 2019), và có hai phiên bản: PhoBERT-base (150M parameters) và PhoBERT-large (350M parameters). Trong báo cáo này, chúng em chỉ sử dụng phiên bản PhoBERT-base và thực hiện *fine-tuning* cho nhiệm vụ sentiment classification trên dữ liệu phản hồi trong lĩnh vực dược phẩm.



Hình 3.1: Tổng quan về kiến trúc của BERT

PhoBERT, cũng như RoBERTa, đều kế thừa và phát triển từ kiến trúc của **BERT** (Bidirectional Encoder Representations from Transformers). Về cơ bản, BERT là một kiến trúc đa tầng gồm nhiều lớp **Bidirectional Transformer Encoder**, cụ thể như sau:

- **Bidirectional:** Khác với các mô hình trước đây chỉ xử lý ngữ cảnh một chiều (từ trái sang phải hoặc ngược lại), BERT đọc toàn bộ câu để nắm bắt ngữ cảnh của từ từ cả hai phía.
- **Transformer Encoder:** Sử dụng các lớp mã hoá Transformer với cơ chế *self-attention* giúp mô hình tập trung vào các phần quan trọng của câu.

BERT được huấn luyện trước (*pre-trained*) thông qua hai nhiệm vụ chính:

- **Masked Language Modeling (MLM):** Dự đoán từ bị che trong câu dựa vào ngữ cảnh xung quanh.
- **Next Sentence Prediction (NSP):** Dự đoán xem câu thứ hai có phải là câu tiếp theo trong văn bản gốc hay không.

Sau khi đã có được cái nhìn tổng quan về PhoBERT rồi, tiếp theo chúng ta sẽ cùng đi sâu vào phân tích xem, để có thể có được một model PhoBERT phục vụ cho tác vụ Classification, thì chúng ta sẽ phải làm những gì

—

3.2 Kiến trúc của BERT

PhoBERT được phát triển dựa trên kiến trúc của BERT — một mô hình nổi bật trong xử lý ngôn ngữ tự nhiên (NLP). Kiến trúc của BERT được thiết kế để nắm bắt thông tin ngữ cảnh toàn diện từ cả hai hướng của câu thay vì chỉ một chiều như các mô hình truyền thống. Cấu trúc chính của BERT bao gồm các thành phần sau:

1. Cấu trúc tổng quan của BERT

BERT (Bidirectional Encoder Representations from Transformers) bao gồm nhiều lớp **Transformer Encoder** được tổ chức theo các khối. Mỗi khối gồm:

- **Embedding Layer:**
  - Biểu diễn từ dưới dạng vector (Word Embeddings).
  - Biểu diễn vị trí (Positional Embeddings) để mô hình hiểu thứ tự từ.
  - Biểu diễn phân đoạn (Segment Embeddings) cho các tác vụ có cặp câu.
- **Transformer Encoder Layers:**
  - *Self-Attention Mechanism:* Cho phép mô hình tập trung vào các từ quan trọng trong câu, nắm bắt ngữ cảnh hai chiều.
  - *Feed-Forward Neural Network (FFN):* Biến đổi phi tuyến để học đặc trưng sâu hơn.
  - *Residual Connection & Normalization:* Giúp ổn định gradient và giữ lại thông tin đầu vào.
- **Output Layer:** Biểu diễn của token [CLS] được dùng làm đầu vào cho các tác vụ như phân loại.

2. Cơ chế Self-Attention

Công thức của self-attention trong BERT:

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

Trong đó:

- Q, K, V: lần lượt là ma trận Query, Key và Value.
- d_k: số chiều của vector khóa.
- *softmax*: chuẩn hóa trọng số để tạo phân phối xác suất.

Self-Attention là cơ chế cho phép mô hình nhìn bao quát toàn bộ chuỗi đầu vào để đánh giá mức độ quan trọng của từng yếu tố, từ đó tập trung có chọn lọc vào những thông tin liên quan nhất nhằm hiểu rõ ngữ cảnh.

3. Masked Language Modeling (MLM)

Trong MLM, một số từ trong câu được thay bằng token `[MASK]`, và mô hình dự đoán từ ban đầu dựa vào ngữ cảnh hai chiều. Điều này giúp BERT học được mối quan hệ giữa các từ trong câu một cách toàn diện.

4. Next Sentence Prediction (NSP)

Nhiệm vụ này yêu cầu mô hình dự đoán xem câu B có phải là câu tiếp theo của câu A trong văn bản gốc hay không rất hữu ích cho các bài toán như hỏi đáp hoặc phân tích đoạn văn. Nhờ đó, nó không chỉ hiểu nội dung đơn lẻ mà còn nhận biết được đâu là nguyên nhân dẫn đến kết quả, sự việc nào diễn ra trước sau theo trình tự thời gian, và phân biệt được khi nào mạch văn đang liền mạch hay đã chuyển sang một chủ đề nội dung hoàn toàn khác

PhoBERT kế thừa toàn bộ kiến trúc này và tinh chỉnh riêng cho tiếng Việt, giúp tối ưu hóa khả năng biểu diễn ngữ nghĩa ngôn ngữ Việt.

3.3 Fine-tuning cho tác vụ Classification

Quá trình tinh chỉnh mô hình **PhoBERT** cho bài toán phân loại cảm xúc gồm các bước chính sau:

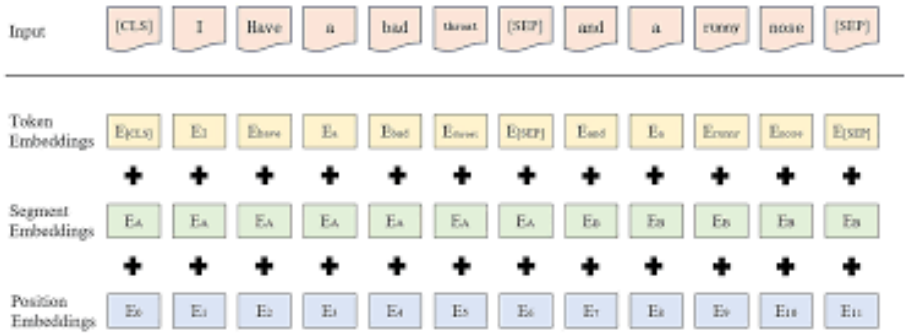
1. Thêm lớp phân loại đầu ra

Một lớp Fully Connected (FC) được thêm vào đầu ra của token `[CLS]` nhằm dự đoán nhãn cảm xúc của toàn bộ đoạn văn bản. Biểu diễn ẩn của token `[CLS]` sau khi qua mô hình PhoBERT sẽ được sử dụng làm đầu vào cho lớp FC này.

2. Tokenization

Chuyển văn bản thô thành các đơn vị nhỏ hơn, được gọi là các token, mà mô hình có thể xử lý. Với BERT, quá trình này bao gồm việc thêm các token đặc biệt và chuyển đổi các từ thành các chỉ số (index) tương ứng trong từ điển của BERT.

1. **Thêm các token đặc biệt:**
    - `[CLS]`: Token đặc biệt được thêm vào đầu mỗi đoạn văn bản, biểu diễn toàn bộ nội dung của câu.
    - `[SEP]`: Token dùng để phân tách các câu hoặc đánh dấu kết thúc đoạn văn bản.



Hình 3.2: Tokenizer

2. **Chuyển đổi thành token:** Sử dụng `tokenizer` của PhoBERT để tách văn bản thành các token con (subword).
3. **Chuyển đổi thành chỉ số (index):** Mỗi token được ánh xạ tới một chỉ số duy nhất trong từ điển của mô hình. Trong mô hình này sử dụng từ điển VinAi.
4. **Padding và Attention Mask:** Để đảm bảo đầu vào có độ dài đồng nhất, các câu ngắn hơn sẽ được bổ sung bằng token `[PAD]`. Đồng thời, `attention mask` là một mảng nhị phân xác định token nào được mô hình chú ý (1) và token nào bị bỏ qua (0).

```
{'text': 'chúc một ngày tốt lành',
 'input_ids': tensor([   0, 3788,   16,   43,  167, 4446,    2,    0,    0,    0]),
 'attention_masks': tensor([1, 1, 1, 1, 1, 1, 1, 0, 0, 0]),
 'targets': tensor(1)}
```

Hình 3.3: Ví dụ về quá trình tokenization trong PhoBERT

3. Huấn luyện mô hình

- **Chuẩn bị dữ liệu:** Chia dữ liệu thành train/validation bằng *k-fold cross validation*.

Hình 3.4: Minh họa cho Cross-Validation

- **DataLoader:** Tải dữ liệu theo batch size để tối ưu hiệu năng huấn luyện.
- **Xây dựng mô hình:** Khởi tạo mô hình SentimentClassifier, định nghĩa hàm loss (criterion) và tối ưu hóa (optimizer). Trong quá trình này, chúng ta sử dụng hàm loss là Cross Entropy Loss và tối ưu hóa bằng thuật toán AdamW.

Hàm mất mát – Cross Entropy Loss: Trong bài toán phân loại, hàm mất mát được sử dụng phổ biến là Cross Entropy Loss, nhằm đo lường mức độ khác biệt giữa phân phối xác suất dự đoán của mô hình và phân phối thực tế của dữ liệu. Công thức tổng quát được biểu diễn như sau:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij} \cdot \log(p_{ij}) \tag{3.1}$$

Trong đó:

- N: Số lượng mẫu trong tập huấn luyện.
- C: Số lượng lớp (nhãn) trong bài toán phân loại.

- y_{ij}: Giá trị thực tế của mẫu i đối với lớp j (1 nếu mẫu i thuộc lớp j, ngược lại là 0).
- p_{ij}: Xác suất mô hình dự đoán rằng mẫu i thuộc lớp j.

Hàm mất mát này khuyến khích mô hình dự đoán xác suất cao hơn cho nhãn đúng và giảm dần xác suất cho các nhãn sai, giúp cải thiện khả năng phân loại chính xác qua từng epoch.

Tối ưu hoá (AdamW): Tham số:

- η: Tốc độ học (learning rate).
- λ: Hệ số Weight Decay .
- β_1, β_2: Các hệ số suy giảm cho momentum.
- ϵ: Hằng số nhỏ để tránh chia cho 0.

Thuật toán thực hiện qua 4 bước:

- **Tính Gradient:** Tính đạo hàm của hàm mất mát đối với trọng số:

$$g_t = \nabla f(\theta_{t-1})$$

- **Cập nhật các Moment:** Tính toán quán tính (m) và độ biến động (v):

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

- **Sửa lỗi chệch (Bias Correction):** Hiệu chỉnh sai số khởi tạo ban đầu:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

- **Cập nhật tham số (Bước quyết định của AdamW):** Đây là điểm khác biệt cốt lõi, AdamW tách biệt phần giảm trọng số (Weight Decay) ra khỏi bước cập nhật thích ứng:

$$\theta_t = \theta_{t-1} - \underbrace{\eta \left(\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \right)}_{\text{Bước cập nhật Adam}} - \underbrace{\eta \lambda \theta_{t-1}}_{\text{Weight Decay tách biệt}}$$

Quy trình Huấn luyện và Tối ưu hóa (Training Loop): Quá trình huấn luyện được thực hiện lặp lại qua nhiều kỷ nguyên (*epochs*) trên tập dữ liệu huấn luyện. Tại mỗi bước lặp (*iteration*), dữ liệu được chia nhỏ thành các lô (*mini-batches*) để tận dụng khả năng tính toán song song của GPU. Chu trình tối ưu hóa diễn ra khép kín gồm 4 giai đoạn chính:

1. **Lan truyền xuôi (Forward Pass):**
   Các tensor đầu vào (bao gồm `input_ids` và `attention_mask`) được đưa qua kiến trúc PhoBERT để trích xuất đặc trưng ngữ cảnh. Tại lớp đầu ra (*Classification Head*), mô hình tính toán các giá trị *logits* , thể hiện mức độ tự tin của mô hình đối với từng nhãn cảm xúc (Tiêu cực, Trung tính, Tích cực).

2. **Tính toán hàm mất mát (Loss Calculation):**
   Kết quả dự đoán được so sánh với nhãn thực tế thông qua hàm mất mát Cross-Entropy Loss. Hàm này đo lường độ lệch giữa phân phối xác suất dự đoán (sau khi đi qua hàm Softmax) và nhãn thực. Giá trị Loss càng thấp chứng tỏ mô hình dự đoán càng chính xác.

3. **Lan truyền ngược (Backward Pass):**
   Hệ thống sử dụng thuật toán lan truyền ngược Backpropagation để tính toán đạo hàm riêng gradients của hàm mất mát đối với từng tham số trong mạng nơ-ron.

4. **Cập nhật trọng số (Optimization step):**
   Dựa trên gradient đã tính toán, thuật toán tối ưu hóa AdamW sẽ thực hiện điều chỉnh các trọng số θ của mô hình nhằm giảm thiểu hàm mất mát. Bên cạnh đó, tốc độ học learning rate được điều chỉnh động thông qua bộ lập lịch Scheduler, giúp mô hình hội tụ nhanh ở giai đoạn đầu và tinh chỉnh chính xác hơn ở các giai đoạn sau. Cuối cùng, gradient được đặt lại về 0 để chuẩn bị cho bước lặp tiếp theo.

4. Tổng kết và Hướng phát triển

4.1 Tổng kết quá trình thực hiện

Tổng kết lại, nghiên cứu đề xuất xây dựng mô hình phân loại cảm xúc văn bản tiếng Việt dựa trên kỹ thuật học chuyển giao (Transfer Learning), với kiến trúc nòng cốt là mô hình ngôn ngữ tiền huấn luyện PhoBERT-base.

Cụ thể, PhoBERT đóng vai trò là bộ mã hóa (Encoder) giúp trích xuất các đặc trưng ngữ nghĩa ngữ cảnh hai chiều từ chuỗi đầu vào. Vector đại diện cho toàn bộ câu (tương ứng với token đặc biệt [CLS]) sau đó được đưa qua một lớp kết nối đầy đủ (Fully Connected Layer) để thực hiện nhiệm vụ phân loại.

Trong quá trình huấn luyện, toàn bộ trọng số của mô hình được tinh chỉnh (fine-tuning) đồng thời nhằm tối ưu hóa hàm mất mát Cross-Entropy Loss. Bên cạnh đó, thuật toán tối ưu hóa AdamW được lựa chọn để cập nhật trọng số nhờ khả năng thích ứng tốc độ học hiệu quả. Để đảm bảo tính khách quan và độ tin cậy của kết quả, nghiên cứu áp dụng chiến lược đánh giá kiểm định chéo K lần (K-fold Cross-validation), qua đó lựa chọn được phiên bản mô hình tối ưu nhất.

4.2 Hướng phát triển

Mặc dù hệ thống đã đạt được những kết quả khả quan bước đầu, nhưng để nâng cao hiệu năng và khả năng ứng dụng thực tế trong quản trị nhân sự, đề tài đề xuất một số hướng phát triển trong tương lai như sau:

- Phân tích cảm xúc dựa trên khía cạnh (Aspect-Based Sentiment Analysis - ABSA): Hiện tại mô hình mới chỉ phân loại cảm xúc chung. Hướng phát triển tiếp theo là xác định cảm xúc gắn với từng khía cạnh cụ thể (ví dụ: thái độ nhân viên, tốc độ phục vụ) để bộ phận nhân sự có cái nhìn chi tiết hơn.

- Áp dụng Active Learning trong gán nhãn dữ liệu: Nghiên cứu áp dụng kỹ thuật Active Learning để mô hình tự đề xuất các mẫu dữ liệu khó cho con người gán nhãn, giúp giảm thiểu chi phí nhân lực mà vẫn tăng độ chính xác.

- Xử lý ngôn ngữ tự nhiên đa phương thức (Multimodal NLP): Mở rộng nguồn dữ liệu đầu vào bao gồm cả hình ảnh hoặc giọng nói từ các video review để có cái nhìn toàn diện hơn.

- Tối ưu hóa mô hình để triển khai: Nghiên cứu các kỹ thuật nén mô hình (Model Distillation) để giảm dung lượng và tăng tốc độ suy diễn (Inference time), cho phép

hệ thống xử lý thời gian thực (Real-time processing).

Tài liệu tham khảo

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, NAACL 2019.

[2] Dat Quoc Nguyen, Anh Tuan Nguyen, *PhoBERT: Pre-trained Language Models for Vietnamese*, Findings of EMNLP 2020.

[3] Quoc-Nam Nguyen et al., *VisoBERT: A Pre-trained Language Model for Vietnamese Social Media Text Processing*, 2023.

[4] Chi Sun et al., *How to Fine-tune BERT for Text Classification?*, 2020.

[5] Nguyen Minh Vu et al., *Pre-training and Fine-tuning ELECTRA Models for Vietnamese NLP Tasks*, 2021.

[6] Lê Hồng Phương, Nguyễn Thị Minh Huyền, Azim Roussanaly, Hồ Tường Vinh, *A Hybrid Approach to Word Segmentation of Vietnamese Texts*, Lecture Notes in Computer Science, 2013.