ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC - KỸ THUẬT MÁY TÍNH

CÁCH TIẾP CẬN HIỆN ĐẠI TRONG XỬ LÝ NGÔN NGỮ TỰ NHIÊN

Bài tập lớn

SENTIMENT ANALYSIS

Giảng viên hướng dẫn: PGS.TS. Quản Thành Thơ
Sinh viên: Nguyễn Văn Hậu 2111167

TP. HỒ CHÍ MINH, THÁNG 4/2025

Mục lục

Tóm tắt nội dung

Báo cáo này trình bày chi tiết việc triển khai và đánh giá các mô hình học sâu khác nhau cho bài toán phân tích cảm xúc trên tập dữ liệu tiếng Việt của cuộc thi VLSP 2016 (Task SA). Mục tiêu là phân loại cảm xúc của văn bản (tích cực, tiêu cực, trung tính) bằng bốn phương pháp tiếp cận riêng biệt: Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), mô hình kết hợp CNN-LSTM và mô hình dựa trên Transformer (PhoBERT). Các embedding Word2Vec tiếng Việt tiền huấn luyện đã được sử dụng cho các mô hình không phải BERT. Mỗi mô hình được huấn luyện, đánh giá và hiệu suất của chúng được so sánh bằng các độ đo tiêu chuẩn như accuracy (độ chính xác), precision, recall, f1-score. Kết quả thực nghiệm cho thấy mô hình dựa trên PhoBERT đạt được hiệu suất cao nhất, chứng minh hiệu quả của việc học chuyển giao với các transformer tiền huấn luyện cho nhiệm vụ này. Các thách thức, cải tiến tiềm năng và phân tích so sánh giữa các mô hình cũng được thảo luận.

1 Giới thiệu

Phân tích cảm xúc (hay còn được gọi là khai thác ý kiến hoặc cảm xúc AI) là việc sử dụng xử lý ngôn ngữ tự nhiên, phân tích văn bản, ngôn ngữ học tính toán và sinh trắc học để xác định một cách có hệ thống, trích xuất, định lượng và nghiên cứu trạng thái cảm xúc và thông tin chủ quan[3]. Các ứng dụng của nó rất đa dạng, từ nghiên cứu thị trường, theo dõi thương hiệu đến phân tích chính trị và hiểu phản hồi của khách hàng. Với sự phát triển của học sâu, các mô hình như CNN, LSTM và Transformer (như BERT) đã cho thấy tiềm năng đáng kể trong việc nắm bắt các mẫu ngôn ngữ phức tạp để phân loại cảm xúc. Bài tập lớn này tập trung vào việc áp dụng các kỹ thuật này cho tiếng Việt, một ngôn ngữ ít tài nguyên hơn so với tiếng Anh, sử dụng một tập dữ liệu chuẩn.

1.1 Bài toán

Bài toán chính được giải quyết là phân loại cảm xúc được thể hiện trong văn bản tiếng Việt từ tập dữ liệu của cuộc thi VLSP (Vietnamese Language and Speech Processing). Mục tiêu là gán nhãn cảm xúc (tích cực, tiêu cực, trung tính) cho mỗi mẫu văn bản một cách chính xác.

1.2 Mục tiêu

Các mục tiêu chính của bài tập lớn:

- Xây dựng và huấn luyện các mô hình học sâu khác nhau để giải quyết bài toán phân loại cảm xúc trên tập dữ liệu của cuộc thi VLSP. Các mô hình bao gồm: Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), mô hình kết hợp CNN và LSTM và mô hình dựa trên kiến trúc Transformer (BERT, cụ thể là PhoBERT).
- Đánh giá hiệu năng của các mô hình đã xây dựng trên tập dữ liệu kiểm thử (test set).
- So sánh kết quả giữa các phương pháp, rút ra nhận xét về ưu nhược điểm của từng cách tiếp cận đối với bài toán và dữ liệu cụ thể này.

1.3 Mô tả tập dữ liệu

Bài tập sử dụng tập dữ liệu được cung cấp trong cuộc thi Vietnamese Language and Speech Processing (VLSP) năm 2016, Task SA (Sentiment Analysis). Dữ liệu bao gồm các bình luận của người dùng về nhiều chủ đề khác nhau.

- Tập huấn luyện (Train set): Gồm 5100 mẫu, được lưu trong file vlsp_sentiment_train.csv.
- Tập kiểm thử (Test set): Gồm 1050 mẫu, được lưu trong file vlsp_sentiment_test.csv.
- Nhãn (Labels): Mỗi bình luận được gán một trong ba nhãn:

  - -1: Tiêu cực (Negative)
  - 0: Trung tính (Neutral)
  - 1: Tích cực (Positive)

  Định dạng: Dữ liệu được lưu dưới dạng file CSV, phân tách bằng dấu tab, với cột đầu tiên là nhãn và cột thứ hai là nội dung bình luận.

2 Phương pháp tiếp cận

2.1 Tiền xử lý dữ liệu

Dữ liệu văn bản thô (cột 'Data') từ cả tập huấn luyện và kiểm thử ban đầu đã trải qua các bước tiền xử lý sau:

- Loại bỏ chữ số: Tất cả các ký tự số bị loại bỏ khỏi văn bản.
- Tách từ tiếng Việt và chuyển đổi chữ thường: Thư viện 'pyvi' được sử dụng để tách từ (ViTokenizer.tokenize) và đồng thời chuyển văn bản thành chữ thường. Kết quả là danh sách các chuỗi văn bản đã được làm sạch và tách từ, sẵn sàng cho bước tokenization của mô hình Word2Vec (với CNN, LSTM) hoặc Transformer (với BERT).
- Mã hóa nhãn dữ liệu:
  - Đối với mô hình CNN, LSTM, CNN+LSTM: Chuyển đổi nhãn (-1, 0, 1) thành dạng one-hot encoding (ví dụ: [1,0,0], [0,1,0], [0,0,1]).
  - Đối với BERT: Sử dụng LabelEncoder của scikit-learn để chuyển nhãn (-1, 0, 1) thành (0, 1, 2).

2.2 Embeddings

Để chuyển đổi văn bản thô thành các biểu diễn số phù hợp cho các mô hình học sâu, hai phương pháp embedding khác nhau tùy thuộc vào kiến trúc mô hình sẽ được áp dụng:

Word Embeddings (Word2Vec): Đối với các mô hình CNN, LSTM và CNN-LSTM, các embedding Word2Vec tiếng Việt tiền huấn luyện được sử dụng.

- Mô hình: CBOW (Continuous Bag-of-Words)
- Nguồn: Cung cấp qua các bài lab (nêu rõ nguồn nếu biết cụ thể hơn)
- Số chiều: 300.
- Cách sử dụng: Các embedding này được tải và tích hợp vào các mô hình (CNN, LSTM, CNN-LSTM) làm trọng số khởi tạo cho lớp Embedding Layer, đặt `trainable=True` để có thể vừa huấn luyện mô hình phân loại cảm xúc vừa cập nhật lại các embeddings (fine-tuning).

Contextual Embeddings (BERT): Đối với mô hình dựa trên BERT, chúng tôi sử dụng các embedding ngữ cảnh từ mô hình tiền huấn luyện đặc biệt cho tiếng Việt:

- Mô hình: **PhoBERT** - Phiên bản tối ưu hóa RoBERTa cho tiếng Việt, hỗ trợ hiểu ngữ cảnh hai chiều (bidirection) [2].
- Nguồn: Tiền huấn luyện bởi VinAI Research, triển khai qua thư viện Hugging Face Transformers.
- Số chiều: 768 (PhoBERT-base) hoặc 1024 (PhoBERT-large) tùy phiên bản.
- Cách sử dụng: Mô hình PhoBERT (vinai/phobert-base) tự động tạo ra các vector biểu diễn ngữ cảnh (contextualized embeddings) cho từng token đầu vào thông qua kiến trúc Transformer của nó. Các embeddings này được tinh chỉnh (fine-tuned) trong quá trình huấn luyện mô hình cho tác vụ phân loại quan điểm.

So sánh:

- Word2Vec phù hợp cho mô hình đơn giản nhưng bị giới hạn bởi tính chất tĩnh của embedding.
- PhoBERT vượt trội trong xử lý đa nghĩa và phụ thuộc ngữ cảnh nhờ cơ chế self-attention, đặc biệt hữu ích cho cảm xúc phức tạp ("phim chán quá!"vs "chán quá phim hay!").
- Quá trình fine-tuning PhoBERT yêu cầu tài nguyên tính toán cao hơn nhưng cho kết quả tối ưu.

2.3 Mô hình CNN

Kiến trúc: Input (sequence of token IDs) -> Lớp Embedding (Word2Vec, 400 dims, trainable) -> Nhiều nhánh Convolution song song:

- Nhánh 1: Conv1D (filter size 4, 100 filters, activation='relu', kernel_regularizer=L2) -> MaxPooling1D (pool size để giảm chiều dài về 1)
- Nhánh 2: Conv1D (filter size 6, 100 filters, activation='relu', kernel_regularizer=L2) -> MaxPooling1D
- Nhánh 3: Conv1D (filter size 8, 100 filters, activation='relu', kernel_regularizer=L2) -> MaxPooling1D

-> Concatenate các output từ các lớp MaxPooling -> Flatten -> Dropout (rate 0.5) -> Lớp Dense cuối cùng (3 units, activation='softmax', kernel_regularizer=L2).

Ý tưởng: CNN có khả năng trích xuất các đặc trưng cục bộ (local features) hiệu quả, tương tự như việc nhận diện các cụm từ n-gram quan trọng (ví dụ: "rất tốt", "không hài lòng") thể hiện cảm xúc. Việc sử dụng nhiều kích thước bộ lọc (filter sizes) khác nhau (4, 6, 8) cho phép mô hình nắm bắt các cụm từ có độ dài khác nhau. MaxPooling giúp chọn ra đặc trưng quan trọng nhất từ mỗi bộ lọc.

2.4 Mô hình LSTM - Bidirectional LSTM

Kiến trúc: Input (sequence of token IDs) -> Lớp Embedding (Word2Vec, 400 dims, trainable) -> SpatialDropout1D (rate 0.2) -> Reshape -> Lớp Bidirectional LSTM (512 units, return_sequences=True) -> Lớp Bidirectional LSTM (256 units, return_sequences=True) -> Lớp Bidirectional LSTM (128 units, return_sequences=False) -> Dropout (rate 0.5) -> Lớp Dense cuối cùng (3 units, activation='softmax', kernel_regularizer=L2(0.02)).

Ý tưởng: LSTM (Long Short-Term Memory) là một loại mạng nơ-ron hồi quy (RNN) được thiết kế để xử lý dữ liệu tuần tự và nắm bắt các phụ thuộc xa (long-range dependencies). Bidirectional LSTM xử lý chuỗi theo cả hai chiều (từ trái sang phải và từ phải sang trái), giúp mô hình hiểu ngữ cảnh tốt hơn dựa trên thông tin từ cả quá khứ và tương lai của một từ trong câu. Việc xếp chồng nhiều lớp LSTM cho phép học các biểu diễn trừu tượng hơn. SpatialDropout1D giúp chống overfitting bằng cách bỏ qua toàn bộ feature maps thay vì các units riêng lẻ.

2.5 Mô hình LSTM kết hợp với Attention

Kiến trúc: Tương tự như mô hình BiLSTM chuẩn, nhưng thêm một cơ chế Attention sau lớp BiLSTM cuối cùng (128 units, return_sequences=True): ... -> Lớp BiLSTM cuối cùng (128 units, return_sequences=True) -> Lớp Dense (units=128, activation='tanh', name='attention_dense_1') -> Lớp Dense (units=1, activation='linear', name='attention_scores') -> Lớp Activation ('softmax', name='attention_weights') -> Lớp Multiply() ([Output BiLSTM cuối, Trọng số Attention]) -> Lớp Lambda (reduce_sum trên chiều thời gian) (Tạo context vector) -> Dropout (rate 0.5) -> Lớp Dense cuối cùng (3 units, activation='softmax', kernel_regularizer=L2(0.01)).

Ý tưởng: Bổ sung cơ chế Attention cho phép mô hình tập trung vào những phần quan trọng nhất của chuỗi đầu vào khi đưa ra dự đoán cuối cùng. Thay vì chỉ sử dụng trạng thái ẩn cuối cùng của LSTM, Attention tính toán một "context vector"bằng cách lấy tổng có trọng số của tất cả các trạng thái ẩn theo thời gian, nơi trọng số thể hiện mức độ quan trọng của từng trạng thái.

2.6 Mô hình CNN kết hợp LSTM

Kiến trúc: Input (sequence of token IDs) -> Lớp Embedding (Word2Vec, 400 dims, trainable) -> Reshape -> Nhiều nhánh Convolution song song:

- Nhánh 1: Conv1D (filter size 3, 100 filters, activation='relu', kernel_regularizer=L2(0.01)) -> MaxPooling1D
- Nhánh 2: Conv1D (filter size 4, 100 filters, activation='relu', kernel_regularizer=L2(0.01)) -> MaxPooling1D
- Nhánh 3: Conv1D (filter size 5, 100 filters, activation='relu', kernel_regularizer=L2(0.01)) -> MaxPooling1D

-> Reshape output của từng nhánh MaxPooling -> Concatenate các output đã reshape -> Dropout (rate 0.3) -> Lớp LSTM (256 units) -> Dropout (rate 0.3) -> Lớp Dense cuối cùng (3 units, activation='softmax', kernel_regularizer=L2(0.01)).

Ý tưởng: Mô hình này kết hợp ưu điểm của cả CNN và LSTM. Các lớp CNN đầu tiên hoạt động như bộ trích xuất đặc trưng cục bộ (n-grams), sau đó các đặc trưng này được đưa vào lớp LSTM để mô hình hóa mối quan hệ tuần tự và ngữ cảnh giữa chúng.

2.7 Mô hình dựa trên BERT - PhoBERT

Kiến trúc: Input (đã được tokenize bởi PhoBERT tokenizer) -> Mô hình PhoBERT (vinai/phobert-base) -> Lớp Dropout (thường có sẵn trong kiến trúc head) -> Lớp Dense cuối cùng (classification head) với 3 units và hàm kích hoạt softmax.

Ý tưởng: Tận dụng sức mạnh của mô hình Transformer đã được tiền huấn luyện (pre-trained) trên một kho dữ liệu tiếng Việt lớn. PhoBERT có khả năng hiểu sâu sắc ngữ nghĩa và ngữ cảnh của văn bản, sau đó chỉ cần tinh chỉnh (fine-tune) một lớp phân loại nhỏ phía trên cho tác vụ cụ thể.

3 Thực nghiệm

3.1 Thiết lập thực nghiệm

Môi trường: Các thử nghiệm được thực hiện trên nền tảng Kaggle Notebook, sử dụng GPU (NVIDIA Tesla T4) để tăng tốc quá trình huấn luyện.

Thư viện chính:

- tensorflow: Xây dựng và huấn luyện các mô hình CNN, LSTM, CNN kết hợp LSTM.
- transformers (từ Hugging Face): Tải, sử dụng tokenizer và mô hình PhoBERT.
- Datasets (từ Hugging Face): Tạo và quản lý tập dữ liệu cho Trainer.
- pyvi: Hỗ trợ tách từ tiếng Việt.
- gensim: Tải mô hình Word2Vec tiền huấn luyện.
- pandas, numpy: Xử lý dữ liệu.
- scikit-learn: Đánh giá mô hình (classification_report, accuracy_score), mã hóa nhãn (LabelEncoder).
- matplotlib, seaborn: Trực quan hóa ma trận nhầm lẫn.

Hyperparameters:

- Mô hình Word2Vec (CNN, LSTM, CNN+LSTM):
  - EMBEDDING_DIM: 400

  - MAX_VOCAB_SIZE: 10000
  - MAX_SEQUENCE_LENGTH: 300
  - Optimizer: AdamW - Adam với weight decay (learning rate 1e-4)
  - Loss: categorical_crossentropy
  - Batch size: 256
  - Epochs: 20 (với Early Stopping, patience=4, theo dõi val_loss)
  - Dropout rate: 0.5
  - Regularization: L2

- Mô hình PhoBERT:
  - Model name: vinai/phobert-base
  - Max sequence length: 256
  - Optimizer: AdamW (mặc định trong TrainingArguments)
  - Learning rate: 1e-5
  - Batch size (train): 16 (tích lũy gradient 2 bước -> effective batch size 32)
  - Batch size (eval): 32
  - Epochs: 5
  - Weight decay: 0.01
  - Warmup ratio: 0.1
  - Mixed precision: fp16=True

3.2 Quá trình huấn luyện

Mô hình Word2Vec: Sử dụng hàm model.fit của Keras. Dữ liệu được chia thành tập huấn luyện và tập validation (tỷ lệ 9010) ngay trong quá trình fit. Sử dụng EarlyStopping để tránh overfitting và dừng huấn luyện khi val_loss không cải thiện. Trọng số mô hình tốt nhất (dựa trên val_loss) được lưu lại.

Mô hình PhoBERT: Sử dụng Trainer của thư viện Transformers. Tập train_dataset và test_dataset (đã qua tokenizer) được cung cấp trực tiếp. TrainingArguments điều khiển quá trình huấn luyện, bao gồm chiến lược đánh giá (eval_strategy="epoch"), lưu mô hình (save_strategy="epoch"), và tự động lưu mô hình tốt nhất dựa trên accuracy (load_best_model_at_end=True, metric_for_best_model="accuracy").

4 Kết quả

Hiệu năng của các mô hình được đánh giá trên tập kiểm thử (test set) gồm 1050 mẫu. Các độ đo chính bao gồm Accuracy, Precision, Recall, và F1-Score (tính trung bình macro).

Mô hình	Accuracy	Precision	Recall	F1-Score
CNN (W2V)	62.00%	0.6545	0.6200	0.5998
LSTM (W2V)	64.29%	0.6436	0.6429	0.6432
LSTM + Attention (W2V)	61.33%	0.6302	0.6133	0.6106
CNN+LSTM (W2V)	61.52%	0.6128	0.6152	0.6115
PhoBERT	**73.14%**	**0.7329**	**0.7314**	**0.7314**

Bảng 1: Kết quả đánh giá các mô hình trên tập kiểm thử VLSP 2016 SA (Precision, Recall, F1 là Macro Average)

PhoBERT: Mô hình dựa trên PhoBERT cho kết quả vượt trội nhất với accuracy đạt 73.14% và các chỉ số F1-score (cả macro và weighted) cũng cao nhất. Điều này khẳng định sức mạnh của các mô hình Transformer tiền huấn luyện và phương pháp fine-tuning cho các tác vụ NLP cụ thể, ngay cả với ngôn ngữ ít tài nguyên như tiếng Việt. Khả năng hiểu ngữ cảnh sâu sắc của PhoBERT giúp nó phân loại cảm xúc chính xác hơn.

Mô hình Word2Vec:

- Mô hình LSTM (64.29% Accuracy) hoạt động tốt hơn so với mô hình CNN (62.00%) và mô hình kết hợp CNN+LSTM (61.52%). Điều này cho thấy việc nắm bắt các phụ thuộc tuần tự và ngữ cảnh dài hạn của LSTM có thể quan trọng hơn việc chỉ trích xuất đặc trưng cục bộ của CNN cho bài toán này.
- Mô hình kết hợp CNN+LSTM không mang lại cải thiện so với LSTM đơn lẻ, thậm chí còn thấp hơn một chút. Có thể kiến trúc cụ thể hoặc siêu tham số chưa được tối ưu hoàn toàn, hoặc việc kết hợp không mang lại lợi ích cộng hưởng như mong đợi trên tập dữ liệu này.

Phân tích lớp: Nhìn vào classification report và confusion matrix của từng mô hình (chi tiết trong các notebook):

- Các mô hình thường gặp khó khăn trong việc phân biệt giữa lớp tiêu cực (0) và trung tính (1), hoặc trung tính (1) và tích cực (2).
- PhoBERT tỏ ra cân bằng hơn giữa các lớp so với các mô hình Word2Vec, thể hiện qua chỉ số F1-score khá đồng đều cho cả 3 lớp (0.74, 0.65, 0.81).
- Mô hình CNN (W2V) khá yếu ở lớp 1 (F1=0.47) và mô hình LSTM (W2V) cũng gặp khó khăn tương tự (F1=0.59). Mô hình CNN+LSTM (W2V) cân bằng hơn một chút (F1=0.59, 0.55, 0.69).

5 Phương pháp cải tiến

Dựa trên kết quả và quá trình thực nghiệm, một số hướng cải tiến tiềm năng bao gồm:

- Tiền xử lý nâng cao: Xử lý các trường hợp phủ định phức tạp, biểu tượng cảm xúc (emoticons/emojis), tiếng lóng, từ viết tắt đặc trưng trong bình luận tiếng Việt.

- Tăng cường dữ liệu (Data Augmentation): Sử dụng các kỹ thuật như back-translation, synonym replacement để tăng kích thước tập huấn luyện, đặc biệt hữu ích cho các mô hình cần nhiều dữ liệu như Transformer.
- Tinh chỉnh siêu tham số: Sử dụng các công cụ tìm kiếm siêu tham số tự động (ví dụ: Keras Tuner, Optuna, Ray Tune) để tối ưu learning rate, dropout rate, số lớp/units, kích thước bộ lọc CNN, kiến trúc LSTM,...

Ngoài ra, để cải thiện hơn nữa hiệu suất phân loại cảm xúc trên tập dữ liệu VLSP 2016, đặc biệt là vượt qua kết quả của PhoBERT, thì hiện nay còn có một phương pháp sử dụng các Large Language Model (LLMs) kết hợp với kỹ thuật Prompting [1]. Các LLM này, với kiến trúc Transformer tiên tiến và được huấn luyện trên lượng dữ liệu khổng lồ, có khả năng hiểu ngữ nghĩa và ngữ cảnh sâu sắc hơn, hứa hẹn mang lại độ chính xác cao hơn. Có hai hướng tiếp cận chính:

5.1 Sử dụng LLM thông qua API

Ý tưởng: Tận dụng sức mạnh của các LLM tiên tiến như Gemini (hoặc các mô hình tương tự như GPT-3.5, 4.0...) mà không cần tự huấn luyện hay triển khai cơ sở hạ tầng phức tạp. Việc phân loại được thực hiện bằng cách gửi văn bản cần phân tích đến API của LLM cùng với một chỉ dẫn (prompt) rõ ràng.

Cách tiếp cận:

- Zero-Shot Prompting: Cung cấp cho LLM chỉ dẫn về tác vụ và yêu cầu nó phân loại cảm xúc của văn bản đầu vào mà không cần cung cấp bất kỳ ví dụ nào về dữ liệu VLSP.
  - Ví dụ:

```
Phân loại cảm xúc của bình luận tiếng Việt sau đây thành một
trong ba loại: Tích cực, Tiêu cực, Trung tính.
Chỉ trả về một nhãn duy nhất.

Bình luận: "[Nội dung bình luận cần phân loại]"

Cảm xúc:
```

Hình 1: Prompt mẫu cho Zero-Shot Learning

- Few-Shot Prompting: Cung cấp cho LLM chỉ dẫn tác vụ cùng với một vài ví dụ (văn bản và nhãn cảm xúc tương ứng từ tập huấn luyện VLSP) ngay trong prompt, giúp LLM "học tại chỗ"(in-context learning) và hiểu rõ hơn yêu cầu cụ thể của bài toán trên dữ liệu này.
  - Ví dụ:

```
Phân loại cảm xúc của các bình luận tiếng Việt sau đây
thành một trong ba loại: Tích cực, Tiêu cực, Trung tính.
```

```
Chỉ trả về một nhãn duy nhất.
---
Ví dụ 1:
Văn bản: "Điện thoại này dùng rất tốt, pin trâu, chụp ảnh đẹp."
Cảm xúc: Tích cực

Ví dụ 2:
Văn bản: "Sách được giao khá nhanh."
Cảm xúc: Trung tính

Ví dụ 3:
Văn bản: "Quán phục vụ quá chậm, đồ ăn lại không ngon."
Cảm xúc: Tiêu cực
---

Bình luận cần phân loại:
Văn bản: "[Nội dung bình luận cần phân loại]"
Cảm xúc:
```

Hình 2: Prompt mẫu cho Few-Shot Learning

Ưu điểm:

- Dễ dàng triển khai, không yêu cầu tài nguyên tính toán lớn để huấn luyện.
- Có thể tiếp cận các mô hình rất mạnh mẽ và cập nhật liên tục.

Nhược điểm:

- Phụ thuộc vào API của bên thứ ba (có thể tốn chi phí, giới hạn tốc độ, lo ngại về bảo mật dữ liệu).
- Hiệu năng phụ thuộc nhiều vào chất lượng của prompt (Prompt Engineering).
- Có thể không tối ưu bằng mô hình được fine-tune trực tiếp trên dữ liệu cụ thể.

5.2 Fine-tuning LLM (self-host)

Ý tưởng: Lấy một mô hình LLM tiền huấn luyện (có thể là mô hình đa ngôn ngữ hỗ trợ tiếng Việt như XLM-RoBERTa, hoặc các LLM tiếng Việt mới hơn nếu có thể truy cập và fine-tune) và tiếp tục huấn luyện (fine-tune) nó trên tập dữ liệu huấn luyện VLSP 2016.

Cách tiếp cận: Sau khi fine-tune xong LLM, tiến hành tạo prompt giống như cách tiếp cận trên.

Ưu điểm:

- Thường cho hiệu năng tốt nhất vì mô hình được chuyên biệt hóa cho tác vụ và dữ liệu cụ thể.
- Toàn quyền kiểm soát mô hình và dữ liệu (nếu tự host).

Nhược điểm:

- Yêu cầu tài nguyên tính toán đáng kể (GPU mạnh, nhiều bộ nhớ).
- Chi phí thiết lập và vận hành có thể cao (nếu tự host).
- Cần đảm bảo đủ dữ liệu huấn luyện chất lượng để fine-tuning hiệu quả.

6 Kết luận

Bài tập lớn Nghiên cứu này đã thực hiện thành công việc xây dựng, huấn luyện và đánh giá một cách hệ thống bốn mô hình học sâu (CNN, LSTM, CNN-LSTM với Word2Vec, và PhoBERT) ứng dụng cho bài toán phân tích cảm xúc tiếng Việt trên tập dữ liệu VLSP 2016. Kết quả nổi bật là hiệu năng vượt trội của PhoBERT (Accuracy 73.14%), chứng minh sức mạnh của kiến trúc Transformer và phương pháp học chuyển giao (transfer learning) trong việc xử lý ngôn ngữ tự nhiên tiếng Việt. Các mô hình truyền thống hơn như LSTM cũng đạt được kết quả khả quan, trong khi CNN và CNN-LSTM tỏ ra kém hiệu quả hơn trong thiết lập này. Báo cáo đã cung cấp phân tích định lượng và định tính về điểm mạnh, điểm yếu của từng mô hình, đồng thời gợi mở các phương pháp cải tiến tiên tiến, đóng góp vào việc lựa chọn và phát triển các giải pháp phân tích cảm xúc hiệu quả cho tiếng Việt.

Tài liệu tham khảo

[1] Tharindu Kumarage, Amrita Bhattacharjee, and Joshua Garland. Harnessing artificial intelligence to combat online hate: Exploring the challenges and opportunities of large language models in hate speech detection, 2024.

[2] Dat Quoc Nguyen and Anh Tuan Nguyen. Phobert: Pre-trained language models for vietnamese, 2020.

[3] Sentiment analysis. Sentiment analysis — Wikipedia, the free encyclopedia, 2025.