ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**
**KHOA TOÁN - TIN HỌC**

---


# BÁO CÁO ĐỒ ÁN THỰC HÀNH

## Môn học: Xử lý ngôn ngữ tự nhiên

Tháng 6 năm 2025

# Bảng phân công và đóng góp

| **Họ và Tên** | **Đặng Minh Phúc** | **Nguyễn Minh Hùng** | **Trịnh Ngọc Mạnh Hùng** |
|---|---|---|---|
| **MSSV** | 22280064 | 21110301 | 22280036 |
| **Vai trò** | Trưởng nhóm | Thành viên | Thành viên |
| **Công việc** | - Tiền xử lý dữ liệu<br>- Huấn luyện mô hình LSTM và GRU<br>- Viết báo cáo<br>- Tổng hợp báo cáo | - Phân tích dữ liệu<br>- Huấn luyện mô hình BERT và XLNet<br>- Viết báo cáo | - Huấn luyện mô hình Logistic Regression và SVM<br>- Viết báo cáo |
| **Đóng góp** | 36% | 32% | 32% |

# 1 Sơ lược đồ án

## 1.1 Giới thiệu vấn đề

Tin giả là mối đe dọa nghiêm trọng đối với xã hội, gây xói mòn niềm tin công chúng, thao túng bầu cử và lan truyền thông tin sai lệch trong các tình huống khủng hoảng. Việc phát hiện tin giả bằng NLP gặp nhiều thách thức, do chúng thường bắt chước ngôn ngữ báo chí chính thống, thiếu dữ liệu nhãn chất lượng và liên tục thay đổi cách diễn đạt để né tránh hệ thống phát hiện. Bên cạnh đó, các yếu tố như ngữ cảnh văn hóa, châm biếm và định kiến trong dữ liệu huấn luyện có thể khiến mô hình đưa ra dự đoán sai lệch. Do vậy, việc xây dựng hệ thống phát hiện cần được thực hiện một cách thận trọng và có ý thức về ngữ cảnh.

Trong báo cáo này, nhóm chúng tôi triển khai và so sánh hiệu quả của các mô hình Học máy truyền thống, Học sâu và Mô hình Ngôn ngữ lớn (LLM) trong nhiệm vụ phân loại tin thật và tin giả.

## 1.2 Bộ dữ liệu

Dữ liệu được thu thập từ nhiều trang tin tức trực tuyến khác nhau như Reuters, The New York Times, The Washington Post, v.v. Nhóm đã chia bộ dữ liệu ban đầu ra thành 3 tập train, test, val với tỉ lệ là 6:2:2.

## 1.3 Cấu trúc phần nộp

Cấu trúc phần nộp gồm

```
|-- report.pdf                # file bao cao hien tai
|-- preprocess.py             # file code tien xu ly du lieu
|-- eda.ipynb                 # file phan tich du lieu
|-- train_and_validate.ipynb  # file huan luyen va so sanh cac
   phuong phap
```

# 2 Phân tích dữ liệu

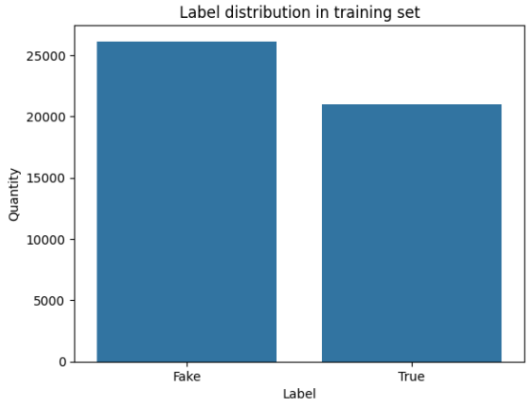
Trước khi huấn luyện mô hình, chúng tôi thực hiện phân tích dữ liệu để hiểu rõ hơn về đặc điểm của bộ dữ liệu, bao gồm phân phối nhãn, độ dài văn bản, và một số thống kê về từ vựng.

## 2.1 Phân phối nhãn

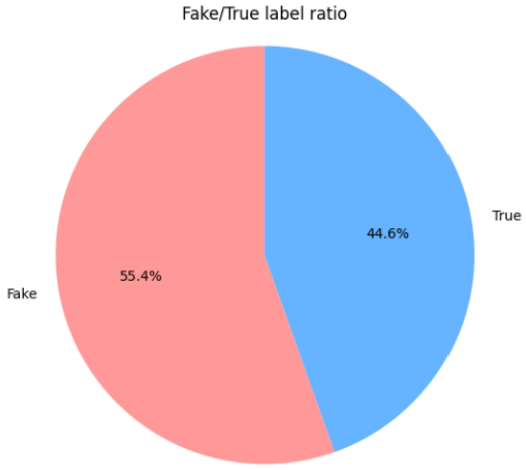
Bộ dữ liệu được chia thành hai nhãn: **0 (tin giả - Fake)** và **1 (tin thật - True)**. Tỷ lệ giữa hai nhãn được thể hiện như sau:

- Nhãn 0 (Fake): **26,145** mẫu — chiếm **55.4%**
- Nhãn 1 (True): **21,009** mẫu — chiếm **44.6%**

Mặc dù có sự chênh lệch nhẹ giữa hai nhãn, dữ liệu vẫn tương đối cân bằng và không cần áp dụng các kỹ thuật xử lý mất cân bằng như *resampling* hay *class weighting*.




(a) Biểu đồ cột phân bố nhãn

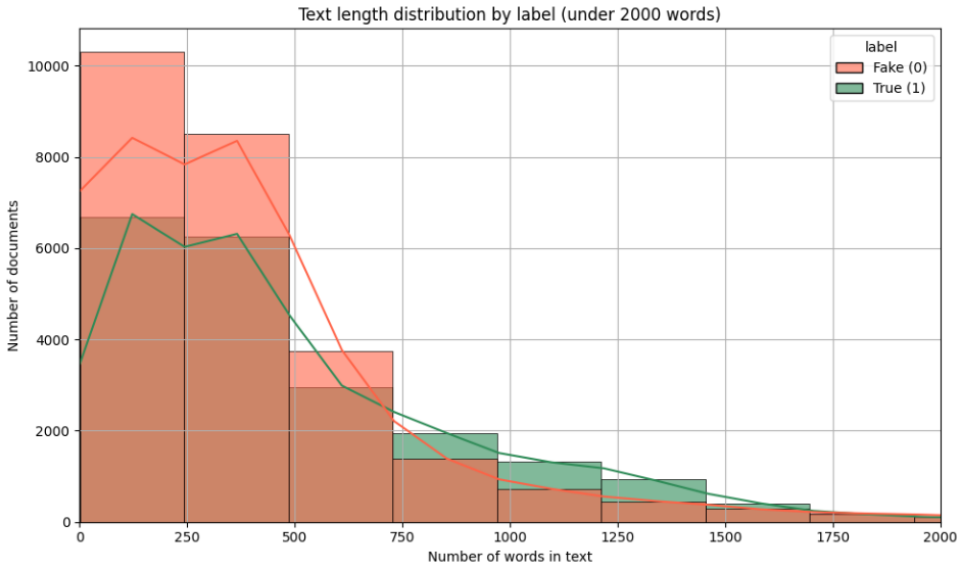



(b) Biểu đồ tròn tỷ lệ nhãn

## 2.2 Phân phối độ dài văn bản

Chúng tôi khảo sát độ dài văn bản (tính theo số từ) trên tập huấn luyện và nhận thấy độ dài phân bố không đều, có nhiều văn bản rất dài (lên đến hơn 24,000 từ). Tuy nhiên, phần lớn văn bản nằm trong khoảng dưới 2,000 từ.

- Với nhãn **tin giả (0)**, độ dài trung bình là 437 từ, trung vị là 325 từ.
- Với nhãn **tin thật (1)** có độ dài trung bình cao hơn (531 từ), trung vị là 387 từ.




Hình 2: Phân phối độ dài văn bản theo nhãn (giới hạn dưới 2000 từ)

Mặc dù có sự khác biệt nhẹ về độ dài giữa hai loại văn bản, biểu đồ cho thấy phân phối không quá lệch, nhờ đó mô hình học không bị thiên lệch theo độ dài văn bản.

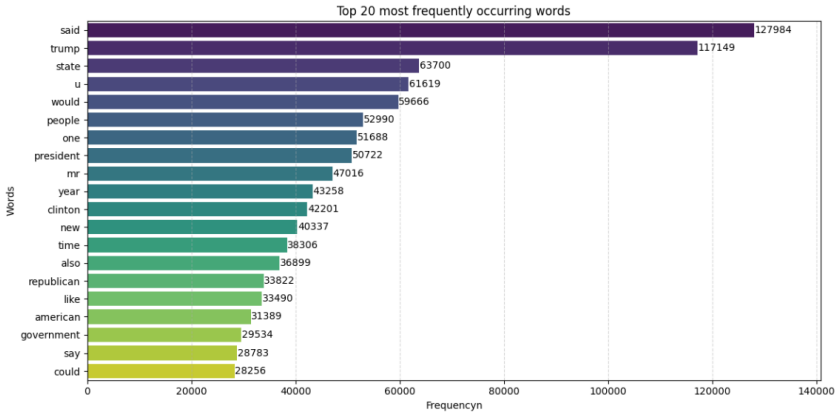
## 2.3 Tần suất từ xuất hiện

Sau khi token hóa tập huấn luyện, chúng tôi tiến hành phân tích tần suất xuất hiện của từ và cụm từ để hiểu rõ hơn về đặc điểm từ vựng trong dữ liệu. Qua thống kê tổng quan, có:

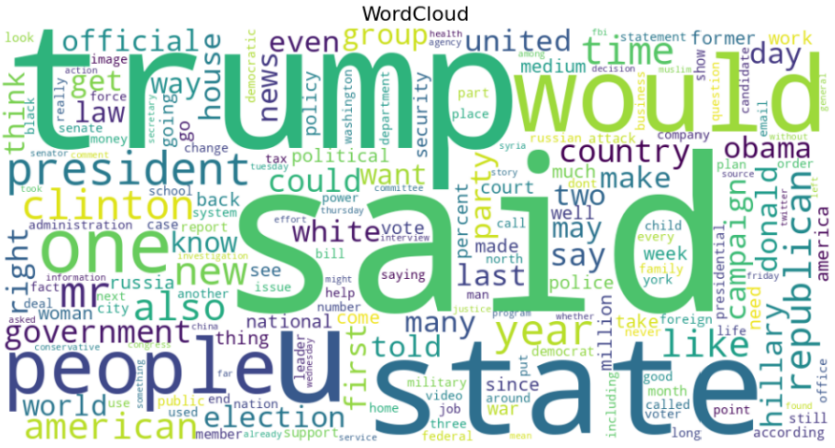
- Tổng số từ (tokens) là **12,628,144**
- Số từ có tần suất xuất hiện $> 0.1\%$ là **108 từ**
- Số từ có tần suất xuất hiện $> 0.01\%$ là **1,821 từ**

Mặc dù tổng số từ rất lớn, chỉ một phần nhỏ từ vựng xuất hiện thường xuyên, cho thấy phân bố từ là không đều.

Chúng tôi tiếp tục phân tích tần suất xuất hiện của từ, Kết quả thu được trong 20 từ phổ biến nhất trong tập dữ liệu, `said` là từ phổ biến nhất tới 127,984 lần xuất hiện, tiếp theo là các từ thường gặp như `trump`, `state`, `would`, `people`, `president`, `clinton`, `republican`, ...



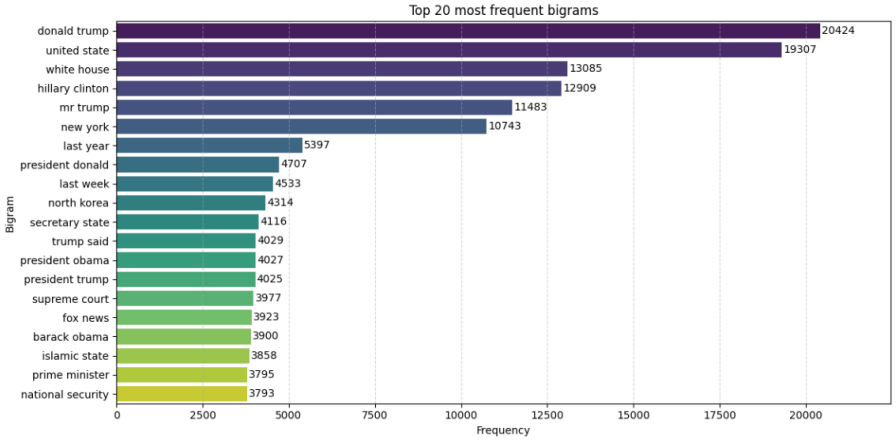

Hình 3: Top 20 từ phổ biến nhất trong tập huấn luyện



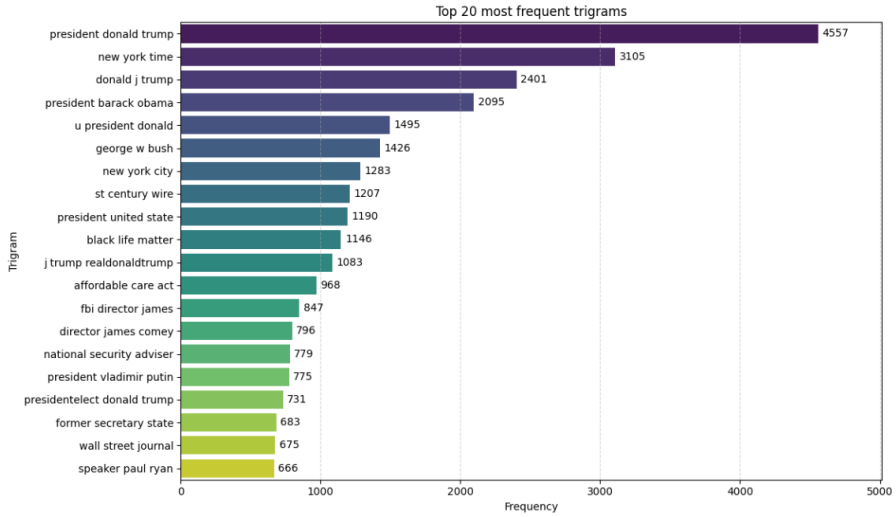

Hình 4: Biểu đồ WordCloud thể hiện các từ phổ biến

Để hiểu rõ hơn về ngữ cảnh, chúng tôi trích xuất các cụm từ (bigrams và trigrams) có tần suất xuất hiện cao. Kết quả phân tích bigrams và trigrams cho được thấy các cụm

từ phổ biến nhất trong dữ liệu như `donald trump`, `united state`, `president donald trump`, ...




Hình 5: Top 20 bigrams phổ biến




Hình 6: Top 20 trigrams phổ biến

Qua các phân tích tần suất từ, chúng tôi đưa ra các nhận xét sau:

- Các từ và cụm từ phổ biến phản ánh rõ chủ đề về chính trị và thời sự Hoa Kỳ, đặc biệt là liên quan đến các nhân vật nổi tiếng như *Donald Trump*, *Barack Obama*, *Hillary Clinton*.
- Tần suất cao của một số cụm từ như `donald trump`, `white house`, `united state` cho thấy nội dung dữ liệu mang tính chất thời sự chính trị mạnh.
- Việc hiểu rõ các cụm từ phổ biến có thể giúp cải thiện việc xử lý tiền xử lý và thiết kế mô hình.

# 3 Xây dựng mô hình

Các mô hình được triển khai bao gồm BERT, XLNet, LSTM, GRU, SVM, Logistic Regression với BERT là mô hình cơ sở.

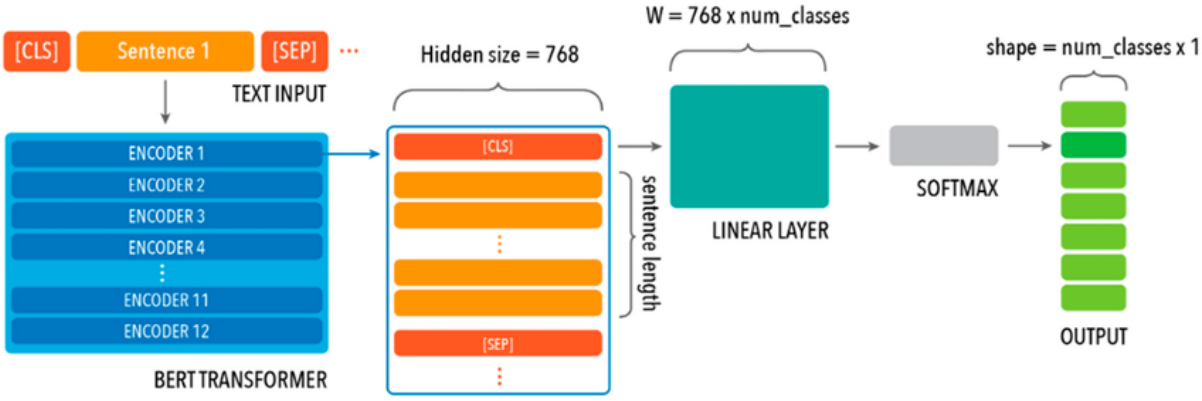
## 3.1 Tiền xử lý dữ liệu

Bước tiền xử lý dữ liệu gồm loại bỏ giá trị bị trùng và giá trị rỗng, và nhúng văn bản. Nhúng văn bản không áp dụng với LLM vì loại mô hình này được tiền huấn luyện trên tập dữ liệu thô, nếu được huấn luyện lại trên dữ liệu được xử lý sẽ khiến mô hình mất đi hiệu suất. Chúng tôi thực hiện các phép biến đổi đơn giản trên văn bản, bao gồm: Chuyển hóa về chữ thường; Loại bỏ số; Loại bỏ các dấu đặc biệt; Loại bỏ stopword; và Chuyển hóa về từ gốc. Tiếp theo, với mô hình Học máy truyền thống, chúng tôi sử dụng TF-IDF để vector hóa từ với `max_features` là 1821; với mô hình Học sâu, chúng tôi sử dụng Tokenizer từ TensorFlow để nhúng từ với `num_words` là 1821.

## 3.2 Mô hình ngôn ngữ lớn

### 3.2.1 BERT và XLNet

BERT (Bidirectional Encoder Representations from Transformers) là một mô hình ngôn ngữ được Google giới thiệu năm 2018, dựa trên kiến trúc Transformer nhưng chỉ sử dụng phần Encoder. Mục tiêu chính của BERT là học biểu diễn ngữ nghĩa cho văn bản bằng cách tận dụng thông tin ngữ cảnh từ cả hai chiều (trái và phải).

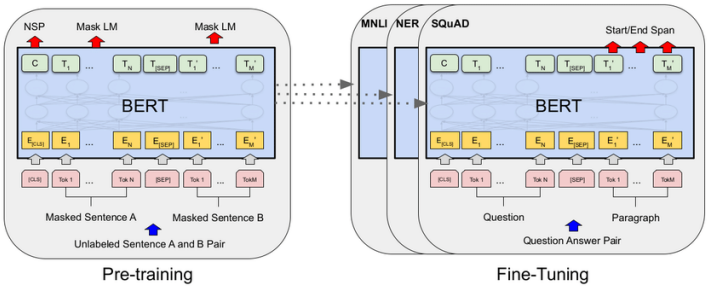



Hình 7: Minh hoạ kiến trúc của BERT

BERT sử dụng kiến trúc Transformer Encoder và học ngữ cảnh bằng cách che một số từ trong câu rồi dự đoán lại. Hai cơ chế chính giúp BERT hoạt động hiệu quả:

- Masked Language Modeling (MLM): BERT thay thế ngẫu nhiên một số từ bằng token [MASK], rồi mô hình phải dự đoán từ bị che dựa vào ngữ cảnh xung quanh. Cách làm này giúp mô hình hiểu mối quan hệ giữa các từ trong câu.
- Next Sentence Prediction (NSP): BERT cũng học cách nhận diện xem hai câu có liên quan với nhau không, giúp mô hình mạnh hơn trong các tác vụ như trả lời câu hỏi và suy luận văn bản.

Sau giai đoạn tiền huấn luyện, BERT có thể được tinh chỉnh (fine-tune) cho nhiều bài toán xử lý ngôn ngữ tự nhiên như phân loại văn bản, nhận dạng thực thể, trả lời câu hỏi, v.v.

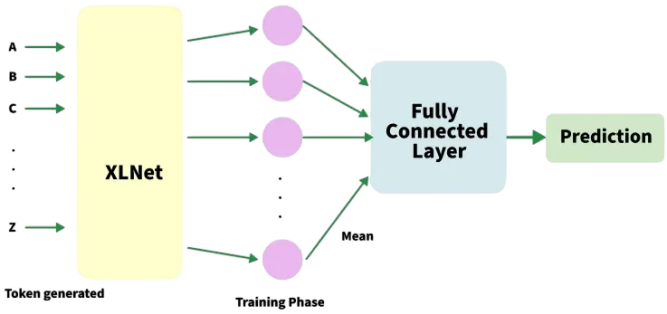



Hình 8: Minh hoạ giai đoạn tiền huấn luyện và tinh chỉnh của BERT

Tuy nhiên, việc sử dụng token [MASK] khiến quá trình huấn luyện và ứng dụng thực tế không hoàn toàn khớp nhau, vì [MASK] không xuất hiện trong văn bản thật. Ngoài ra, các từ bị che được dự đoán độc lập, nên mô hình khó nắm bắt mối quan hệ giữa nhiều từ bị che cùng lúc.

XLNet (2019) kế thừa Transformer-XL và được thiết kế để khắc phục một số hạn chế của BERT. Thay vì che từ như BERT, XLNet sử dụng Permutation Language Modeling (PLM) – học ngữ cảnh từ nhiều thứ tự khác nhau của câu, cho phép mô hình tận dụng toàn bộ thông tin mà không làm thay đổi dữ liệu gốc.

Ngoài ra, nhờ Transformer-XL, XLNet có khả năng ghi nhớ ngữ cảnh dài hơn, giúp xử lý tốt các văn bản phức tạp và dài. Không bị giới hạn bởi token [MASK], XLNet duy trì sự phù hợp giữa quá trình huấn luyện và ứng dụng thực tế, giúp nó hoạt động hiệu quả hơn trong các bài toán yêu cầu suy luận sâu.




Hình 9: Minh hoạ kiến trúc của XLNet

Nhìn chung, BERT và XLNet đều là những mô hình mạnh mẽ dựa trên kiến trúc Transformer, nhưng có cách tiếp cận khác nhau để học ngữ cảnh. BERT sử dụng cơ chế che từ để học ngữ cảnh hai chiều thông qua phương pháp Masked Language Modeling, trong khi XLNet áp dụng Permutation Language Modeling để tận dụng toàn bộ thông tin mà không cần che từ. Mỗi mô hình có ưu điểm riêng: BERT đơn giản và dễ huấn luyện, còn XLNet có khả năng mô hình hóa ngữ cảnh dài hiệu quả hơn và khắc phục một số hạn chế của BERT trong giai đoạn tiền huấn luyện.

### 3.2.2 Thí nghiệm

Chúng tôi sử dụng hai mô hình tiền huấn luyện là `bert-base-uncased` và `xlnet-base-cased` từ thư viện `transformers` của HuggingFace. Cả hai đều có kiến trúc gồm 12 tầng (layers), 768 chiều ẩn (hidden size), và 12 đầu attention, với khoảng 110 triệu tham số. Mỗi mô hình được nối thêm một lớp phân loại tuyến tính (linear layer) để phân biệt giữa hai nhãn: *Fake* và *Real*.

Mô hình `bert-base-uncased` được huấn luyện trên văn bản tiếng Anh đã chuyển thành chữ thường (lower-cased), trong khi `xlnet-base-cased` giữ nguyên phân biệt chữ hoa/chữ thường (cased) và áp dụng cơ chế attention tự hồi tiếp theo thứ tự hoán vị (permutation-based autoregressive) đặc trưng của kiến trúc XLNet.

Chúng tôi tiến hành fine-tune hai mô hình trên tập dữ liệu phân loại tin thật/giả với các siêu tham số như sau:

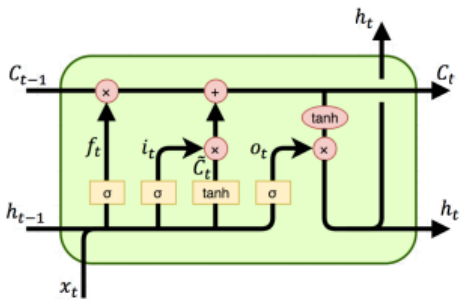
| Tham số | BERT | XLNet |
|---|---|---|
| Max Length | 64 | 64 |
| Batch Size | 8 | 8 |
| Learning Rate | 2e-5 | 2e-5 |
| Epochs | 3 | 3 |

Việc huấn luyện được thực hiện trên GPU, với các siêu tham số giống nhau nhằm đảm bảo tính công bằng trong so sánh. Mô hình được theo dõi bằng độ chính xác (accuracy) và độ mất mát (loss) trên tập validation để tránh overfitting. Sau khi huấn luyện, cả hai mô hình đều được đánh giá trên tập test để so sánh hiệu năng.

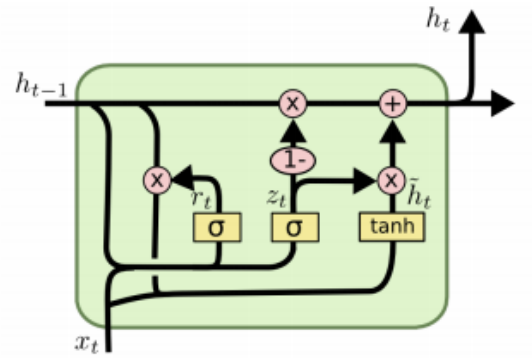
## 3.3 Mô hình học sâu

### 3.3.1 LSTM và GRU

LSTM (Long Short-Term Memory) và GRU (Gated Recurrent Unit) là hai kiến trúc cải tiến của mạng nơ-ron hồi tiếp (RNN), được thiết kế nhằm giải quyết nhược điểm vanishing gradient của RNN truyền thống khi xử lý chuỗi dài. Cả hai mô hình này đều sử dụng cơ chế cổng để điều khiển luồng thông tin, từ đó cho phép mô hình ghi nhớ hoặc quên thông tin một cách có chọn lọc qua các bước thời gian.




(a) Kiến trúc của LSTM




(b) Kiến trúc của GRU

LSTM hoạt động dựa trên ba cổng chính: cổng quên (forget gate), cổng đầu vào (input gate), và cổng đầu ra (output gate). Cổng quên quyết định thông tin nào từ trạng thái

nhớ trước đó sẽ bị loại bỏ; cổng đầu vào quyết định thông tin mới nào cần được ghi nhớ vào trạng thái nhớ; và cổng đầu ra kiểm soát thông tin nào sẽ được xuất ra làm đầu ra ẩn tại thời điểm hiện tại. Nhờ vào trạng thái nhớ riêng biệt (ký hiệu là $C_t$), LSTM có khả năng ghi nhớ thông tin trong thời gian dài, rất phù hợp với các chuỗi có mối quan hệ dài hạn.

GRU là một biến thể đơn giản hơn của LSTM, với chỉ hai cổng: cổng cập nhật (update gate) và cổng đặt lại (reset gate). Cổng cập nhật kiểm soát mức độ giữ lại thông tin cũ và thêm vào thông tin mới, trong khi cổng đặt lại điều chỉnh ảnh hưởng của trạng thái trước lên trạng thái hiện tại. GRU không có trạng thái nhớ riêng biệt như LSTM mà chỉ sử dụng trạng thái ẩn ($h_t$), giúp giảm số lượng tham số và tăng tốc độ huấn luyện. Mặc dù cấu trúc đơn giản hơn, GRU vẫn đạt hiệu quả rất cao trên nhiều bài toán thực tế.

Về tổng thể, LSTM thường hoạt động tốt hơn trong các chuỗi dài và phức tạp, còn GRU thường được ưa chuộng khi yêu cầu mô hình nhỏ gọn, huấn luyện nhanh và dữ liệu không quá dài. Cả hai mô hình đều đóng vai trò quan trọng trong các ứng dụng xử lý ngôn ngữ tự nhiên, phân tích chuỗi thời gian, và nhiều lĩnh vực khác trong học sâu.

### 3.3.2 Thí nghiệm

Chúng tôi tiến hành chạy thí nghiệm với hai GPU Tesla 4 với các tham số sau:

| **Parameter** | **LSTM** | **GRU** |
|---|---|---|
| Loss Function | BCE | BCE |
| Optimizer | RMSprop | RMSprop |
| Input Dim | 1821 | 1821 |
| Embed Dim | 64 | 128 |
| Hidden Dim | 512 | 256 |
| Learning Rate | $5 \cdot 10^{-4}$ | $5 \cdot 10^{-4}$ |
| Epochs | 12 | 7 |

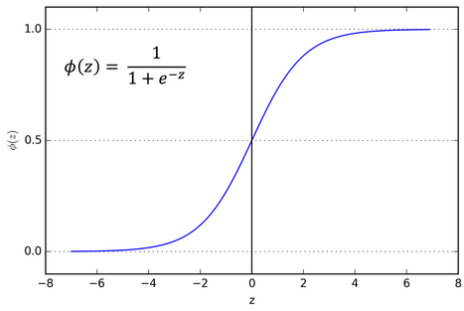
Việc chọn ra được các tham số trên là nhờ việc chạy nhiều lần mô hình và chọn các tham số cho kết quả cao nhất.

## 3.4 Mô hình học máy

### 3.4.1 Logistic Regression

Logistic Regression là một thuật toán học máy có giám sát cơ bản trong bài toán phân loại nhị phân. Thuật toán được thiết kế để dự đoán xác suất thuộc về một lớp cụ thể. Nguyên lý hoạt động của Logistic Regression là sử dụng hàm Sigmoid để chuyển đổi giá trị đầu ra của một hàm tuyến tính (ở đây là tổ hợp tuyến tính của các đặc trưng đầu vào) thành một giá trị nằm trong khoảng từ 0 đến 1. Công thức của hàm sigmoid là:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$




Hình 11: Hàm Sigmoid

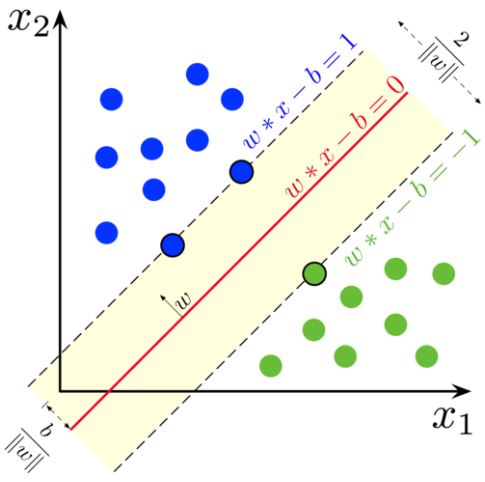
Trong đó, $z = w^T x + b$, với $w$ là vector trọng số, $x$ là vector đầu vào, và $b$ là hệ số điều chỉnh (bias). Mô hình sẽ học cách điều chỉnh các tham số sao cho đầu ra dự đoán (xác suất) gần với nhãn thực tế nhất. Khi cần phân loại, một ngưỡng (threshold) (thường là 0.5) được sử dụng để xác định lớp dự đoán: nếu xác suất từ mức ngưỡng trở lên thì gán vào lớp 1, ngược lại gán vào lớp 0.

Về mặt ý nghĩa hình học, Logistic Regression tìm kiếm một siêu phẳng (hyperplane) tốt nhất để phân tách không gian đặc trưng (feature space) thành hai vùng tương ứng với hai lớp. Ranh giới phân loại này là tập hợp các điểm mà tại đó xác suất là 0.5, tức là nghiệm của phương trình $w^T x + b = 0$.

### 3.4.2 Support Vector Machine - SVM

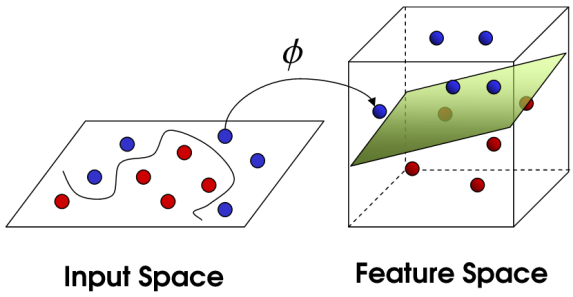
SVM là một thuật toán phân loại dựa trên nguyên lý tìm kiếm siêu phẳng tốt nhất để phân chia dữ liệu thành hai lớp khác nhau.

Ý tưởng chính của thuật toán là tìm một siêu phẳng sao cho khoảng cách từ siêu phẳng đó đến các điểm dữ liệu gần nhất của mỗi lớp (gọi là margin) là lớn nhất. Điều này giúp mô hình có khả năng tổng quát tốt hơn khi gặp dữ liệu mới.




Hình 12: Cách SVM chia biên

Các điểm dữ liệu nằm trên biên của margin được gọi là support vectors - chúng là những điểm quan trọng nhất, những điểm quyết định vị trí của siêu phẳng. Khi dữ liệu không phân tách tuyến tính, SVM sử dụng Kernel trick để ánh xạ dữ liệu từ không gian gốc lên không gian có số chiều cao hơn, nơi có thể tìm được siêu phẳng phân tách tuyến tính.




Hình 13: Minh hoạ Kernel trick

Một vài Kernel trick phổ biến:

- Linear: $K(x_i, x_j) = x_i \cdot x_j$
- Đa thức bậc $p$: $K(x_i, x_j) = (x_i \cdot x_j + r)^p$
- Gaussian: $K(x_i, x_j) = e^{-\gamma||x_i - x_j||^2}$
- Sigmoid: $K(x_i, x_j) = tanh(\gamma x_i \cdot x_j + r)$

### 3.4.3 Thí nghiệm

Chúng tôi tiến hành chạy GridSearch để điều chỉnh các siêu tham số trên Intel(R) Xeon(R) CPU @ 2.20GHz. Kết quả tìm được ở bên dưới:

| Model | C | Class_Weight | Dual | Loss | Penalty |
|---|---|---|---|---|---|
| LinearSVM | 1 | None | True | Hinge | L2 |

| Model | C | Class_Weight | Penalty | Solver |
|---|---|---|---|---|
| Logistic Regression | 10 | None | L2 | Lbfgs |

# 4 Kết quả

| Model | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|
| BERT | 0.9829 | 0.9794 | 0.9818 | 0.9806 |
| XLNet | 0.9856 | 0.9830 | 0.9844 | 0.9837 |
| LSTM | 0.7831 | 0.7837 | 0.7920 | 0.7879 |
| GRU | 0.7686 | 0.7516 | 0.8135 | 0.7813 |
| SVM | 0.9312 | 0.9312 | 0.9312 | 0.9312 |
| LR | 0.9309 | 0.9309 | 0.9310 | 0.9309 |

Bảng so sánh hiệu suất các mô hình cho thấy rằng XLNet đạt kết quả cao nhất trên tất cả các chỉ số, với Precision = 0.9856, Recall = 0.9830, F1-score = 0.9844 và Accuracy = 0.9837. Theo sát là mô hình cơ sở BERT, với các chỉ số cũng rất ấn tượng, cho thấy khả năng biểu diễn ngữ nghĩa mạnh mẽ của các mô hình Transformer. Các mô hình truyền thống như SVM và Logistic Regression (LR) đạt F1-score xấp xỉ 0.931, cho thấy hiệu quả

tốt trong khi yêu cầu ít tài nguyên hơn. Trong khi đó, các mô hình tuần tự như LSTM và GRU thể hiện hiệu suất thấp hơn đáng kể (F1 lần lượt là 0.7920 và 0.8135), điều này phản ánh hạn chế của RNN trong việc xử lý ngữ cảnh dài so với các mô hình dựa trên Attention. Tổng thể, các mô hình Transformer tỏ ra vượt trội và là lựa chọn phù hợp cho các tác vụ phân loại văn bản hiện đại.

Mặc dù LSTM và GRU chưa đạt hiệu suất cao như các mô hình Transformer, tuy nhiên vẫn còn một số hướng cải thiện có thể giúp nâng cao khả năng của các kiến trúc tuần tự này. Thứ nhất, việc kết hợp cơ chế Attention vào LSTM/GRU (ví dụ như mô hình Attention-based BiLSTM) có thể giúp mô hình tập trung tốt hơn vào các thông tin ngữ nghĩa quan trọng trong câu, từ đó cải thiện khả năng xử lý ngữ cảnh dài. Thứ hai, sử dụng các kỹ thuật tiền huấn luyện từ ngữ liệu lớn như ELMo hoặc GloVe để tạo embedding có chất lượng cao hơn cũng giúp tăng hiệu quả biểu diễn đầu vào. Thứ ba, áp dụng kiến trúc song song nhiều tầng (stacked) hoặc hai chiều (Bidirectional) cho LSTM/GRU có thể cải thiện khả năng học ngữ cảnh cả trước và sau. Cuối cùng, việc kết hợp các kỹ thuật regularization như dropout, batch normalization hoặc fine-tuning cẩn thận cũng có thể giảm overfitting và tối ưu hóa hiệu suất mô hình. Những cải tiến này tuy không thể vượt qua các mô hình Transformer trong đa số trường hợp, nhưng có thể là lựa chọn hợp lý khi tài nguyên hạn chế hoặc trong các hệ thống yêu cầu độ trễ thấp.

# 5 Tài liệu tham khảo


- Fake News Classification: Past, Current, and Future
- BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
- LSTM: A Search Space Odyssey
- Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling
- XLNet: Generalized Autoregressive Pretraining for Language Understanding
- Detection of Online Fake News Using N-Gram Analysis and Machine Learning Techniques