

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**-oOo-**



**ĐỒ ÁN CUỐI KỲ  
CS231 – Xử lý ngôn ngữ tự nhiên**

**Đề tài: Sentiment Analysis – Hope Detection with BERTology Models  
And Data Augmentation**

**Giảng viên lý thuyết: Nguyễn Trọng Chính  
Giảng viên thực hành: Đặng Văn Thìn, Nguyễn Đức Vũ**

Nhóm sinh viên thực hiện:

Bùi Hồng Sơn – 22521246

Nguyễn Minh Sơn – 22521254

Đông Minh Quân – 22521176

*Thành phố Hồ Chí Minh, ngày 20 tháng 6 năm 2024*

# Mục Lục

Chương 1: Giới thiệu bài toán .....	3
1.1. Đề tài.....	3
1.2. Mục tiêu đề tài .....	4
1.3. Tác dụng thực tế - Lý do chọn đề tài .....	4
Chương 2: Dữ liệu / Ngữ liệu .....	4
2.1. Thu thập dữ liệu .....	4
2.2. Phân loại nhãn dữ liệu: .....	6
2.3. Thống kê dữ liệu .....	6
2.4. Mẫu dữ liệu .....	8
2.5. Đánh giá dữ liệu.....	8
Chương 3: Phương pháp / Methodology .....	9
3.1. Sơ đồ hóa phương pháp nghiên cứu đề tài.....	9
3.2. Các phương pháp tiền xử lý dữ liệu .....	9
3.2.1. Phương pháp tiền xử lý dữ liệu đơn giản (Simple preprocessing) .....	10
3.2.2. Phương pháp tiền xử lý dữ liệu phức tạp (Enhanced) .....	11
3.3. Phương pháp Data Augmentation [6].....	13
3.4. BERTology models and Model selection.....	14
Chương 4: Cài đặt mô hình và thực nghiệm .....	15
4.1. Cài đặt hệ thống.....	15
4.2. Cài đặt các phương pháp tiền xử lý dữ liệu .....	16
4.2.1. Phương pháp tiền xử lý dữ liệu đơn giản.....	16
4.2.2. Phương pháp tiền xử lý dữ liệu phức tạp.....	16
4.3. Cài đặt mô hình và notebook dành cho quá trình huấn luyện Binary Classification .....	21
4.3.1. Thiết lập môi trường tính toán và đảm bảo tính tái hiện .....	21
4.3.2. Import Train and Validation datasets .....	22
4.3.3. Thiết lập mô hình BERT .....	23
4.3.4. Inference + Evaluation .....	27
4.3.5. Đánh giá trên tập kiểm thử.....	29
4.4. Cài đặt mô hình và notebook dành cho quá trình huấn luyện Multiclass Classification .....	30
4.5. Kết quả thử nghiệm.....	30
4.5.1. Kết quả mô hình Binary Classification trên tập kiểm thử: .....	30
4.5.2. Kết quả mô hình Multiclass Classification trên tập kiểm thử: .....	31
4.6. Mô hình SVM sử dụng GridSearch và TfidfVectorizer ở mức độ cơ bản: .....	32
4.6.1. Kết quả phân Binary Classification: .....	32

4.6.2. Kết quả phần Multiclass Classification: .....	33
4.6.3. Tổng kết .....	34
Chương 5. Kết luận.....	35
5.1. Các điểm nổi bật rút ra từ quá trình thực hiện đề tài:.....	35
5.2. Đóng góp của đề tài .....	36
5.3. Hạn chế và hướng phát triển .....	36
5.4. Tổng kết. ....	36
Các tài liệu tham khảo: .....	37

## Chương 1: Giới thiệu bài toán

### 1.1. Đề tài

**Tên đề tài: Sentiment Analysis – Hope Detection with BERTology Models And Data Augmentation (Phân tích cảm xúc – Phát hiện hy vọng với các mô hình BERT và Data Augmentation)**

Đề tài này tập trung vào phân tích cảm xúc với nhiệm vụ phát hiện hy vọng từ dữ liệu văn bản trên các nền tảng mạng xã hội, cụ thể là các bài đăng và bình luận trên Twitter bằng tiếng Anh. Nhu cầu hiểu rõ hơn về

cách con người biểu đạt hy vọng, mong muốn, và kỳ vọng trên các nền tảng mạng xã hội đã thúc đẩy phát triển đề tài này.

## 1.2. Mục tiêu đề tài

- Phân loại nhị phân/ Binary classification: Xác định liệu một đoạn văn bản có chứa nội dung thể hiện hy vọng hay không, với 2 nhãn (labels) là “Hope” và “Not Hope”.
- Phân loại đa lớp/ Multiclass classification: Phân loại đoạn văn bản thành các loại hy vọng cụ thể hơn bao gồm “Generalized Hope”, “Realistic Hope”, “Unrealistic Hope”, “Not Hope”.

## 1.3. Tác dụng thực tế - Lý do chọn đề tài

Dự án này không chỉ giúp xác định các biểu đạt của hy vọng trên mạng xã hội mà còn mở ra khả năng ứng dụng trong nhiều lĩnh vực khác như phân tích tâm lý học, cải thiện dịch vụ khách hàng, và hỗ trợ nghiên cứu xã hội. Việc sử dụng các mô hình tiên tiến và các kỹ thuật xử lý dữ liệu hiện đại là một bước tiến quan trọng trong việc hiểu và phản hồi cảm xúc của người dùng trên mạng xã hội.

Tại em lựa chọn đề tài này nhằm mục đích là thử nghiệm và so sánh hiệu quả của các mô hình BERT trên dữ liệu từ mạng xã hội, đồng thời nghiên cứu và bước đầu tiếp xúc học hỏi về bài toán Sentiment Analysis trong những lĩnh vực Xử lý ngôn ngữ tự nhiên.

# Chương 2: Dữ liệu / Ngữ liệu

## 2.1. Thu thập dữ liệu

Dữ liệu được lấy từ bộ dữ liệu được chuẩn bị cho cuộc thi Hope at IberLEF 2024 [1] [2]:

- Link cuộc thi:  
[https://codalab.lisn.upsaclay.fr/competitions/17714#learn\\_the\\_details](https://codalab.lisn.upsaclay.fr/competitions/17714#learn_the_details)
- Link dataset kaggle:  
<https://www.kaggle.com/datasets/hongsonuit/dataset-cs221/data>  
<https://www.kaggle.com/datasets/hongsonuit/dataset-cs221->

Dữ liệu (datasets) sử dụng được thu thập nhằm phục vụ cho các nghiên cứu về phân tích cảm xúc và phát hiện hy vọng từ các nền tảng mạng xã hội. Việc hiểu được các biểu hiện hy vọng trong các bài đăng trên mạng xã hội không chỉ có ý nghĩa trong việc phân tích tâm lý người dùng mà còn có thể ứng dụng trong nhiều lĩnh vực khác như hỗ trợ cộng đồng, quảng cáo và tiếp thị xã hội hay là cải thiện dịch vụ khách hàng.

Dữ liệu bao gồm các tweet tiếng Anh, được thu thập trong khoảng thời gian từ tháng 01/2022 đến tháng 06/2022. Dữ liệu này chứa các tweet đề cập đến các khía cạnh khác nhau của hy vọng, từ hy vọng chung chung đến những hy vọng cụ thể hơn.

**Phương hướng thu thập:** Quá trình thu thập bắt đầu bằng việc truy xuất 50,000 tweet gần nhất trong khoảng thời gian từ tháng 01/2022 đến tháng 06/2022. Sau đó, một đợt thu thập bổ sung với 50,000 tweet khác được thực hiện, sử dụng các từ khóa liên quan đến hy vọng như “hope”, “aspire”, “believe”, “expect”, “wish”, và các biến thể của chúng. Các từ khóa này được kiểm tra kỹ lưỡng bởi người bản ngữ để đảm bảo không bỏ sót các từ liên quan được sử dụng trong ngữ cảnh tương tự.

**Phương pháp thu thập:** Dữ liệu được thu thập thông qua API của Twitter và các công cụ web scarping để đảm bảo tính toàn diện và độ chính xác cao trong việc truy xuất các tweet chứa các từ khóa liên quan đến hy vọng. Quá trình lọc và tiền xử lý sau đó đã loại bỏ các tweet trùng lặp, các tweet có ít hơn 10 từ, và các tweet retweet để giữ lại các tweet gốc, độc đáo và có giá trị nội dung cao.

**Số lượng và nội dung:** Sau quá trình lọc, bộ dữ liệu đã giảm xuống còn dưới 30,000 tweet cho mỗi ngôn ngữ. Trong đó, một mẫu ngẫu nhiên gồm 10,000 tweet tiếng Anh đã được chọn để gán nhãn và sử dụng cho nghiên cứu. Các tweet này được đánh giá và phân loại dựa trên mức độ và tính thực tế của hy vọng mà chúng biểu hiện, bao gồm các nhãn “Generalized Hope”, “Realistic Hope”, “Unrealistic Hope”, “Not Hope”.

**Cấu trúc datasets:**

- Train set: 6192 samples
- Validation set: 1032 samples
- Test set: 1032 samples

## 2.2. Phân loại nhãn dữ liệu:

Các nhãn dữ liệu thuộc Binary Classification:

- **Hope:** Những tweet thể hiện hoặc nhắc đến sự hy vọng
- **Not Hope:** Những tweet không thể hiện bất kỳ sự hy vọng, kỳ vọng hoặc mong muốn nào

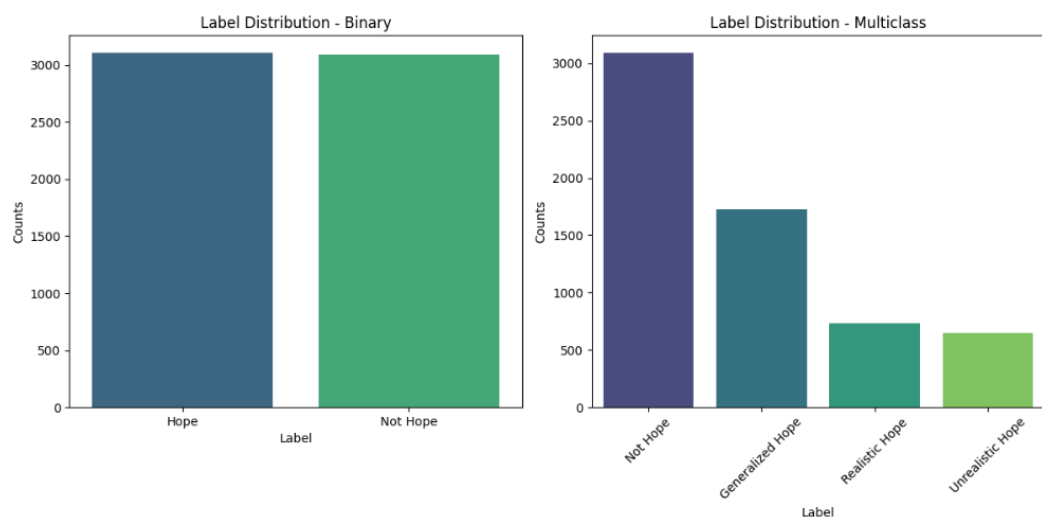
Các nhãn dữ liệu thuộc Multiclass Classification:

- **Generalized Hope:** Được thể hiện như một sự hy vọng chung và lạc quan mà không hướng đến một sự kiện hoặc kết quả cụ thể nào [3] [4] [5].
- **Realistic Hope:** Là mong đợi một điều gì đó hợp lý, có ý nghĩa và có khả năng xảy ra [4]
- **Unrealistic Hope:** Thường là mong muốn một điều gì đó trở thành hiện thực, mặc dù khả năng xảy ra là rất thấp hoặc gần như không thể [5]
- **Not Hope:** Những tweet không thể hiện sự hy vọng

## 2.3. Thống kê dữ liệu

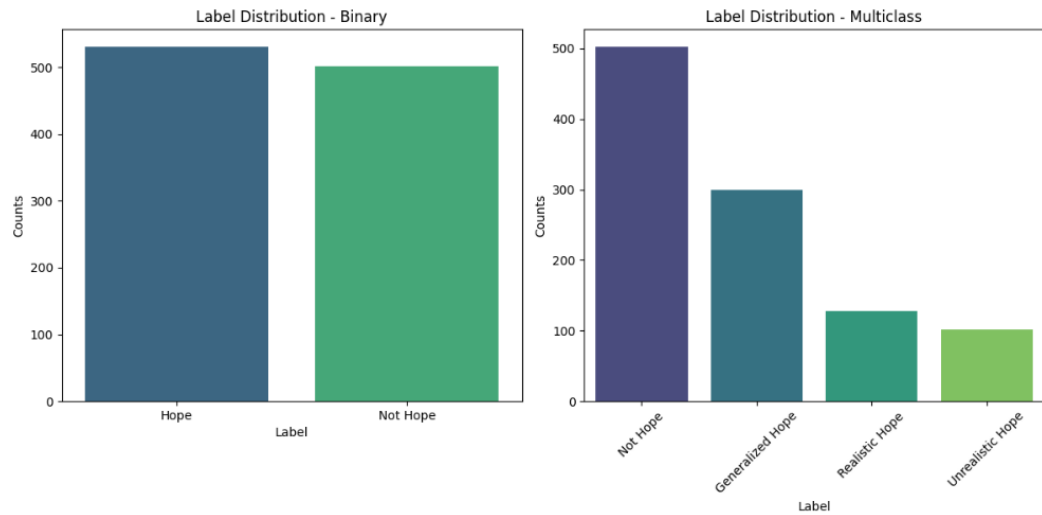
Train dataset tweet samples:

- Max length: 632
- Min length: 52
- Average length: 183
- Number of tokens: 204365
- Number of vocabulary: 33156
- Class Distribution:



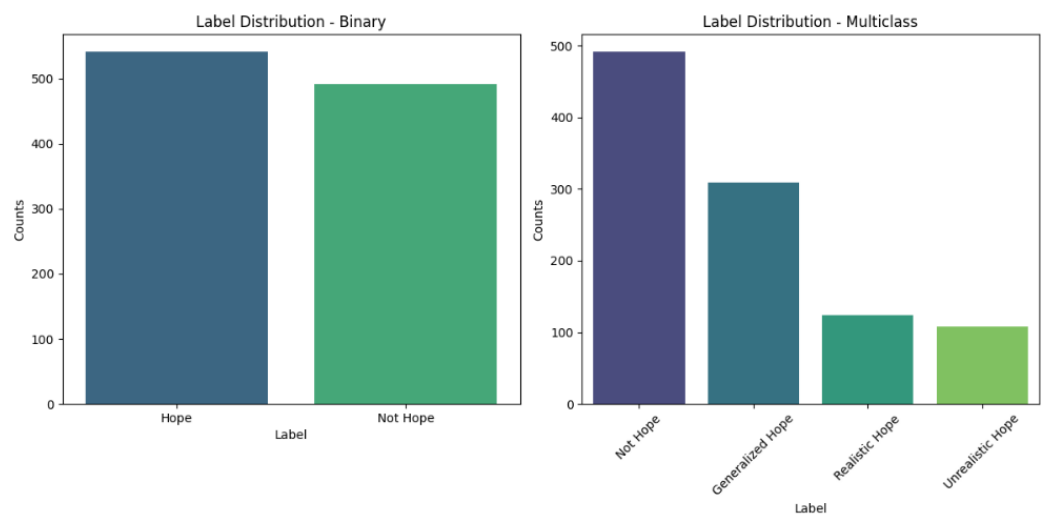
Validation dataset tweet samples:

- Max length: 474
- Min length: 63
- Average length: 177
- Number of tokens: 33098
- Number of vocabulary: 8908
- Class Distribution:



Test dataset tweet samples:

- Max length: 423
- Min length: 63
- Average length: 186
- Number of tokens: 34814
- Number of vocabulary: 9105
- Class Distribution:



## 2.4. Mẫu dữ liệu

**Tweet:** "#USER# #USER# I'm really liking this project, let's work together. Your company trusted believe me. And Congratulations on the launch..best wishes for future projects @AhmedSh27723386 @KdMokabbir @ShakilK70421550"

**Binary Label:** Hope

**Multiclass Label:** Realistic Hope

**Tweet:** "#USER# Oh shit really? I would hope they'd shed some more light on what happened to him in the future"

**Binary Label:** Hope

**Multiclass Label:** Generalized Hope

**Tweet:** "i aspire to have the level of delusion to believe those thing 1 and thing 2 halloween costumes look better than the new version u cannot be serious 🤪  
#URL#"

**Binary Label:** Hope

**Multiclass Label:** Unrealistic Hope

**Tweet:** "Place your thoughts in the comments section, folks, cos' I ain't got the answer. ͇\_ (˘) ͇"

**Binary Label:** Not Hope

**Multiclass Label:** Not Hope

## 2.5. Đánh giá dữ liệu

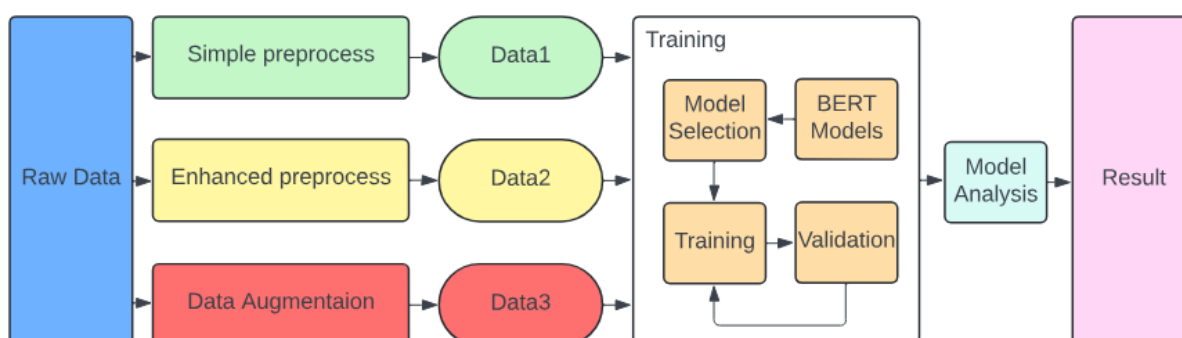
Dữ liệu thu thập được sử dụng trong cuộc thi, nên về mặt khách quan, dữ liệu thu thập đúng, hợp lệ, và gắn nhãn đúng.

Kiểm tra một cách chủ quan đối với một vài mẫu dữ liệu, tụi em xác nhận thêm là các mẫu dữ liệu được gắn nhãn chính xác.



## Chương 3: Phương pháp / Methodology

### 3.1. Sơ đồ hóa phương pháp nghiên cứu đề tài



Mô hình chung, chúng em thực hiện fine-tuning với các mô hình ngôn ngữ đã được huấn luyện trước. Đồng thời nghiên cứu các bước tiền xử lý dữ liệu ảnh hưởng đến hiệu suất của mô hình như thế nào. Điều này là do dữ liệu xuất phát từ nền tảng mạng xã hội, nơi mà việc tiền xử lý đúng cách hoặc đầy đủ có thể cải thiện đáng kể hiệu suất tổng thể của mô hình. Thêm vào đó, chúng em sử dụng các phương pháp Data Augmentation trên dữ liệu để tăng cường thêm dữ liệu dùng cho việc huấn luyện. Đầu tiên, tụi em sẽ sử dụng các mô hình Transformers BERT để huấn luyện trên tập dữ liệu chỉ được tiền xử lý đơn giản, sau đó đánh giá kết quả và lựa chọn ra mô hình cho ra kết quả tốt nhất trong quá trình thí nghiệm. Tiếp theo tụi em áp dụng nhiều các phương pháp xử lý, tiền xử lý dữ liệu khác nhau lên bộ dữ liệu là Tiền xử lý dữ liệu phức tạp, và áp dụng phương pháp Data Augmentation lên bộ dữ liệu huấn luyện. Sau đó đánh giá xem, với phương pháp tiền xử lý dữ liệu phức tạp hơn như lần này thì có cho ra kết quả mô hình khả quan hơn hay không. Đồng thời thực nghiệm phương pháp Data Augmentation đã tìm hiểu để kiểm tra xem phương pháp phổ biến này ảnh hưởng đến quá trình huấn luyện mô hình như thế nào.

### 3.2. Các phương pháp tiền xử lý dữ liệu

Tiền xử lý đề cập đến các phép biến đổi được áp dụng cho dữ liệu của chúng ta trước khi chúng ta đưa nó vào quá trình huấn luyện và đánh giá. Tiền xử lý dữ liệu là bước quan trọng và không thể thiếu trong quá trình phân tích dữ liệu và xây dựng mô hình học máy. Nó giúp cải thiện chất

lượng dữ liệu, tối ưu hóa hiệu suất mô hình và đảm bảo rằng kết quả phân tích chính xác và đáng tin cậy, Việc thực hiện tiền xử lý dữ liệu đúng cách không chỉ giúp tiết kiệm thời gian và tài nguyên mà còn tăng cường khả năng ra quyết định dựa trên dữ liệu.

Đối với dữ liệu văn bản: trong các bài toán xử lý ngôn ngữ tự nhiên, tiền xử lý văn bản có thể bao gồm các phương pháp như loại bỏ dấu câu, chuyển đổi văn bản thành chữ thường, và loại bỏ từ dừng.

Bằng cách thực hiện tiền xử lý dữ liệu một cách cẩn thận và kỹ lưỡng, có thể cải thiện đáng kể hiệu suất và độ tin cậy của các phân tích và mô hình dự đoán của mình.

### 3.2.1. Phương pháp tiền xử lý dữ liệu đơn giản (Simple preprocessing)

Đối với phần này, chúng em thực thực hiện tiền xử lý dữ liệu với hai mức độ : Đơn giản và phức tạp. Các kỹ thuật sử dụng trong Phương pháp tiền xử lý dữ liệu đơn giản (Simple preprocessing):

- **Loại bỏ khoảng trắng ở đầu và cuối văn bản:**  
Khoảng trắng thừa ở đầu và cuối văn bản không mang lại thông tin giá trị và có thể gây ra sự không nhất quán trong việc xử lý dữ liệu. Ví dụ, các từ “hello” và “ hello ” sẽ được coi là khác nhau nếu không loại bỏ khoảng trắng.
- **Loại bỏ dấu câu:**  
Dấu câu như dấu chấm (.), dấu phẩy (,), dấu chấm than(!),... thường không cung cấp thông tin hữu ích gì nhiều trong bài toán Xử lý ngôn ngữ tự nhiên và có thể được loại bỏ để tập trung vào các từ và ngữ nghĩa chính. Tuy nhiên, dấu gạch dưới ( ) được giữ lại trong trường hợp này
- **Loại bỏ khoảng trắng thừa giữa các từ:**  
Trong văn bản, có thể có nhiều khoảng trắng giữa các từ do lỗi nhập liệu hoặc quá trình xử lý trước đó, việc loại bỏ các khoảng trắng giúp làm sạch và chuẩn hóa văn bản, đảm bảo rằng các từ được phân tách một cách nhất quán

Mẫu dữ liệu đã áp dụng phương pháp tiền xử lý đơn giản:

**Raw text:** "#USER# #USER# I'm really liking this project, let's work together. Your company trusted believe me. And Congratulations on the launch..best wishes for future projects @AhmedSh27723386 @KdMokabbir @ShakilK70421550"

**Preprocessed text:** USER USER Im really liking this project lets work together Your company trusted believe me And Congratulations on the launchbest wishes for future projects AhmedSh27723386 KdMokabbir ShakilK70421550"

**Raw text:** "#USER# Good morning, Bud! 🥰 Another good decision from SCOTUS this morning. The light is still bright and things are hopeful. Have a wonderful day, boo 🥰"

**Preprocessed text:** "USER Good morning Bud 🥰 Another good decision from SCOTUS this morning The light is still bright and things are hopeful Have a wonderful day boo 🥰"

Các phương pháp tiền xử lý được sử dụng có tác dụng làm sạch và chuẩn hóa văn bản, giúp loại bỏ các yếu tố không cần thiết như khoảng trắng thừa và dấu câu. Điều này rất quan trọng để cải thiện chất lượng và tính nhất quán của dữ liệu văn bản trước khi tiến hành các bước xử lý tiếp theo như phân tích ngữ nghĩa, phân loại văn bản, hoặc xây dựng mô hình học máy.

### 3.2.2. Phương pháp tiền xử lý dữ liệu phức tạp (Enhanced)

Trong phần này, chúng em thiết kế phương pháp tiền xử lý dữ liệu phức tạp hơn nhằm thí nghiệm xem nếu sử dụng càng nhiều hoặc càng ít biện pháp tiền xử lý dữ liệu hơn thì có cho ra kết quả khả quan hơn không. Các kỹ thuật bổ sung:

- **Mini-handling methods:** loại bỏ token '#USER#' và token '#URL#'. Những token này không mang lại thông tin giá trị cho quá trình phân tích ngữ nghĩa văn bản. Việc loại bỏ các token này giúp làm sạch văn bản và tập trung vào nội dung chính mà không bị ảnh hưởng bởi các yếu tố nhiễu.
- **Lower Casing:** Chuyển văn bản thành chữ thường. Trong văn bản, chữ hoa và chữ thường có thể đại diện cho cùng một từ, nhưng việc phân biệt giữa chúng có thể gây ra sự phức tạp không cần thiết. Ví dụ, từ "Apple" và "apple" sẽ được coi là khác nhau. Chuyển đổi tất

cả các ký tự thành chữ thường giúp đồng nhất hóa dữ liệu và giảm thiểu sự nhầm lẫn, đồng thời giúp các thuật toán học máy hoạt động hiệu quả hơn do số lượng từ duy nhất được giảm xuống.

- **Remove @username mentions:** loại bỏ các văn bản, tag nhắc đến tên người dùng trong đoạn, thường được biểu thị bằng ký tự '@' theo sau là một chuỗi các ký tự. Đây là một bước quan trọng trong quá trình tiền xử lý dữ liệu văn bản từ các nền tảng mạng xã hội như Twitter, nơi việc đề cập đến người dùng rất phổ biến nhưng thường không cung cấp thông tin quan trọng cho các bài toán. Thay vào đó chúng có thể gây nhiễu và làm tăng độ phức tạp của văn bản. Việc loại bỏ giúp làm sạch văn bản và giảm thiểu sự phân tán thông tin không cần thiết.
- **ULR handling method:** loại bỏ các đường dẫn URL bằng cách sử dụng biểu thức chính quy để xác định và thay thế các URL này bằng một chuỗi rỗng.
- **Emojies handling method:** Loại bỏ các biểu tượng cảm xúc và URL từ văn bản để làm sạch và tập trung vào nội dung chính của bản văn. Các biểu tượng cảm xúc (emojies) thường không mang lại giá trị ngữ nghĩa cần thiết. Ngược lại còn làm tăng độ phức tạp của văn bản và gây nhiễu cho quá trình phân tích. Tương tự cũng sử dụng biểu thức chính quy để xác định và thay thế các phần tử này bằng một chuỗi rỗng.
- **Full form transformation:** Loại bỏ các từ viết tắt để làm rõ ý nghĩa của văn bản, giúp cải thiện độ chính xác của các bài toán phân tích văn bản như phân tích cảm xúc, phân loại văn bản. Các từ viết tắt có thể có nhiều dạng khác nhau, gây khó khăn cho việc phân tích. Bằng cách thay thế chúng bằng các từ đầy đủ, văn bản trở nên đồng nhất hơn
- **Remove Punctuation & others:** loại bỏ tất cả các dấu câu từ văn bản và làm cho dữ liệu trở nên dễ xử lý hơn, giảm độ phức tạp của văn bản. Loại bỏ dấu câu giúp giảm số lượng các từ khác nhau trong từ vựng, từ đó giảm thời gian và tài nguyên cần thiết cho việc huấn luyện mô hình. Đồng thời giúp đảm bảo tính nhất quán trong dữ liệu.
- **Lemmatization:** Chuẩn hóa các từ về dạng cơ bản nhất của chúng sử dụng Wordnet của thư viện nltk. Điều này giúp giảm số lượng từ vựng cần xử lý và tăng hiệu quả của các mô hình NLP.

- **Final Lower Casing:** Sau một loạt các thao tác tiền xử lý dữ liệu, chúng em tiến hành chuyển hóa các từ vựng thành dạng bình thường một lần cuối cùng

Mẫu dữ liệu đã áp dụng phương pháp tiền xử lý dữ liệu phức tạp:

**Raw text:** ""#USER# #USER# I'm really liking this project, let's work together. Your company trusted believe me. And Congratulations on the launch..best wishes for future projects @AhmedSh27723386 @KdMokabbir @ShakilK70421550""

**Preprocessed text:** "im really liking this project lets work together your company trusted believe me and congratulations on the launchbest wishes for future projects"

**Raw text:** "#USER# Good morning, Bud! 🥰 Another good decision from SCOTUS this morning. The light is still bright and things are hopeful. Have a wonderful day, boo 🥰"

**Preprocessed text:** "good morning bud another good decision from scotus this morning the light is still bright and things are hopeful have a wonderful day boo"

### 3.3. Phương pháp Data Augmentation [6]

Data Augmentation là một phương pháp quan trọng trong học máy (Machine learning) và xử lý dữ liệu nhằm mở rộng và cải thiện chất lượng dữ liệu huấn luyện bằng cách tạo ra các biến thể mới từ dữ liệu gốc mà vẫn giữ được nhãn hoặc thông tin cần thiết. Điều này giúp cải thiện hiệu suất của mô hình, giảm thiểu hiện tượng overfitting và làm tăng tính đa dạng của dữ liệu

**Sử dụng LLM (Large Language Model) để tạo thêm dữ liệu văn bản:** như GPT-4 hoặc Gemini có khả năng hiểu và tạo ra văn bản một cách tự nhiên, do đó, chúng có thể được sử dụng để tạo ra dữ liệu văn bản mới một cách hiệu quả và linh hoạt. Trong đề tài này, tụi em sử dụng **Paraphrasing (Diễn đạt lại)** – sử dụng LLM là Gemini để tạo ra các câu có ý nghĩa tương tự nhưng với cách diễn đạt khác nhau.

**Lợi ích của việc sử dụng LLM trong Data Augmentation:**

- Tạo ra dữ liệu phong phú và đa dạng, giúp mô hình học máy học có hiệu quả hơn.
- Giảm thiểu hiện tượng overfitting và giúp mô hình học máy hoạt động tốt hơn trên dữ liệu kiểm thử.

- Tiết kiệm thời gian, tạo ra dữ liệu nhanh chóng và dễ dàng mà không cần tốn nhiều thời gian công sức thu thập dữ liệu mới.
- Bằng việc Prompting một cách linh hoạt, LLM có thể dễ dàng điều chỉnh để tạo ra dữ liệu phù hợp với các yêu cầu cụ thể của từng bài toán.

Mẫu dữ liệu được diễn đạt lại sử dụng LLM (Large Language Model):

**Original text:** ""#USER# #USER# I'm really liking this project, let's work together. Your company trusted believe me. And Congratulations on the launch..best wishes for future projects @AhmedSh27723386 @KdMokabbir @ShakilK70421550""

**Generated text:** "Excited about this project! Let's collaborate. Your company's trust means the world to me. Congrats on the launch! Wishing you all the best for future endeavors. #hopeful"

**Original text:** "#USER# Schools are forbidden from initiating or sponsoring religious activities, including prayer, but religious groups are permitted to meet on school grounds after school, and students can pray to whatever or whomever they want at any time of day, as long as they do it privately. FYI"

**Generated text:** "Schools can't start or sponsor religious activities like prayer, but religious groups can meet on school grounds after school. Students can pray to whoever they want, whenever they want, as long as they do it privately and don't mention hope. #ReligiousFreedom #SchoolPrayer"

### 3.4. BERTology models and Model selection

Để xử lý 2 mục tiêu chính của đề tài chúng em đề ra là: Phân loại nhị phân(Binary Classification) và Phân loại nhãn đa lớp (Multiclass Classification). Chúng em tìm hiểu và chọn lựa ra các mô hình Transformers BERT đã được huấn luyện trước (pre-trained models), sau đó fine-tune và sử dụng vào đề tài của chúng em. Các mô hình học máy Transformers BERT được sử dụng:

- **XLM-R-base** [7]: Đây là một mô hình ngôn ngữ mạnh mẽ có khả năng xử lý các nhiệm vụ trên 100 ngôn ngữ khác nhau. Nó sử dụng một kỹ thuật gọi là học tự giám sát, trong đó mô hình phân tích một tập dữ liệu khổng lồ (2,5 TB dữ liệu CommonCrawl đã lọc) mà không có sự can thiệp của con người. Điều này cho phép XLM\_R học từ một lượng văn bản công khai khổng lồ, sử dụng một quá trình tự động để tạo ra cả ví dụ huấn luyện và nhãn từ dữ liệu thô. Trong đề tài này, em sử dụng XLM-R base model (Tham số vừa

đủ, tài nguyên đáp ứng, không yêu cầu nhiều tài nguyên như Large model)

- **DeBERTa** [8]: Chúng em áp dụng DeBERTa-v3-base, một phiên bản cải tiến của DeBERTa để kiểm tra xem liệu chúng em có nhận được kết quả tốt hơn khi sử dụng DeBERTa, một mô hình ngôn ngữ dựa trên Transformers được thiết kế để cải thiện các mô hình BERT và RoBERTa với kỹ thuật: Cơ chế chú ý không phân tách và bộ giải mã mặt nạ được cải tiến
- **Twitter-XLM-roBERTa** [9]: Đây là một mô hình XLM-RoBERTa. Được huấn luyện trên gần 200 triệu tweet bao gồm tám ngôn ngữ (Tiếng Ả Rập, Tiếng Anh, Tiếng Pháp, Tiếng Đức, Tiếng Hindi, Tiếng Ý, Tiếng Tây Ban Nha và Tiếng Bồ Đào Nha). Nó có thể xác định cảm xúc tiêu cực, tích cực hoặc trung tính trong các bài đăng trên mạng xã hội. Mặc dù nó đã được tiền huấn luyện trong các ngôn ngữ cụ thể này, nhưng nó có thể hiểu được cảm xúc trong các ngôn ngữ khác. Chúng em quyết định sử dụng mô hình này để kiểm tra xem liệu nó có hiệu quả khi phân loại các nhãn khác nhau của các văn bản trên Mạng xã hội hay không.

Bằng cách áp dụng đa dạng những mô hình này, chúng em không chỉ cải thiện kết quả của cuối cùng của đề tài mà còn khảo sát khả năng mở rộng việc ứng dụng của mô hình ngôn ngữ trong các bối cảnh thực tế phức tạp. Từ đó, chúng em có thể cung cấp những phân tích chính xác và đáng tin cậy hơn trong lĩnh vực phân tích cảm xúc trong văn bản.

## Chương 4: Cài đặt mô hình và thực nghiệm

### 4.1. Cài đặt hệ thống

Chúng em triển khai quá trình huấn luyện chính của mình với sự hỗ trợ của thư viện Hugging Face Transformers. Tất cả các mô hình đều được thiết lập để huấn luyện với 10 epoch và tốc độ học (learning rate) được đặt là  $2e-4$  cho các mô hình cơ bản. Tùy vào mô hình mà chúng em chọn kích thước batch là 32 hoặc 16, nhưng phần lớn là 16. Các siêu tham số của các mô hình được tinh chỉnh lặp lại dựa trên tập Validation. Phần lớn quá trình huấn luyện mô hình của tụi em được thực hiện trên Kaggle, và bộ gia tốc P100 được chọn để tăng tốc độ huấn luyện. Về phần Tokenizer

sẽ tạo ra với độ dài là 512. Đối với tất cả các thí nghiệm của chúng em, chúng em đặt random seed mặc định là 42. Hai phương pháp tiền xử lý dữ liệu đưa ra đều được áp dụng chung cho cả 2 phần Binary Classification và Multiclass Classification. Chi tiết về cài đặt môi trường và mô hình huấn luyện sẽ được trình bày sau.

## 4.2. Cài đặt các phương pháp tiền xử lý dữ liệu

### 4.2.1. Phương pháp tiền xử lý dữ liệu đơn giản

Đầu tiên cần import thư viện “string” và “re”:

- Thư viện “string” cung cấp các chuỗi và tiện ích liên quan đến việc thao tác với các ký tự, bao gồm các chuỗi đại diện cho các ký tự và dấu câu thông thường.
- Thư viện “re” dành cho việc xử lý các biểu thức chính quy (regular expressions), cho phép thực hiện các thao tác phức tạp với chuỗi văn bản.

```
import string, re
def preprocessing_text(text):
    text = text.strip()
    text = text.translate(text.maketrans("", "",
string.punctuation.replace("_", "")))
    text = re.sub('\s+', ' ', text).strip()
    return text
```

Tác dụng các hàm cài đặt:

- **text = text.strip()**: Loại bỏ khoảng trắng đầu và cuối văn bản
- **text = text.translate()**: Loại bỏ dấu câu khỏi văn bản sử dụng “string.punctuation” để lấy ra chuỗi các ký tự dấu câu chuẩn. Sau đó dùng .maketrans để tạo một bảng dịch để loại bỏ tất cả các dấu câu ngoại trừ dấu gạch dưới (\_)
- **text = re.sub()**: loại bỏ khoảng trắng thừa giữa các từ.

### 4.2.2. Phương pháp tiền xử lý dữ liệu phức tạp

a) Mini handling methods:

```
def removal(text):
    text = text.replace('#USER#', '')
    text = text.replace('#URL#', '')
```



```
text = text.lower()
```

```
return text
```

b) Lower casing

```
#Lower casing
```

```
def lowercase(text):
```

```
    return text.lower()
```

c) Remove @username mentions

```
def remove_mentions(text):
```

```
    mention_regex = r"@[\w+]"
```

```
    return re.sub(mention_regex, "", text)
```

d) URL handling

```
#Removes URL data
```

```
def remove_url(data):
```

```
    url_clean= re.compile(r"https://[S+|www\.|S+)"
```

```
    data=url_clean.sub(r'',data)
```

```
    return data
```

e) Emojis handling

```
#Removes Emojis
```

```
def remove_emoji(data):
```

```
    emoji_clean= re.compile("[
```

```
        u"\U0001F600-\U0001F64F" # emoticons
```

```
        u"\U0001F300-\U0001F5FF" # symbols &
```

```
pictographs
```

```
        u"\U0001F680-\U0001F6FF" # transport & map
```

```
symbols
```

```
        u"\U0001F1E0-\U0001F1FF" # flags (iOS)
```

```
        u"\U00002702-\U000027B0"
```

```
        u"\U000024C2-\U0001F251"
```

```
    "]+", flags=re.UNICODE)
```

```
    data=emoji_clean.sub(r'',data)
```

```
    url_clean= re.compile(r"https://[S+|www\.|S+)"
```

```
    data=url_clean.sub(r'',data)
```

```
    return data
```

Sau đó, làm một lần xử lý với emojis nữa để đảm bảo tất cả emojis đã được xử lý:

```
# Replace Emoji with <emoji>. NO with "
```

```

import emoji
def emoji(text):
    #text = emoji.demojize(text, language='en').replace(':', ' ')
    emoji_regex = r"[\u0000-\uD7FF\uE000-\uF8FF\uFB00-\uFE0F\uFEFF-\uFFFF]"
    #return re.sub(emoji_regex, "<emoji>", text)
    return re.sub(emoji_regex, "", text)

```

f) Full form transformation

**## In this case, we will be replacing some abbreviated pronouns with full forms (example:'you've'->you have')**

```

def remove_abb(data):
    data = re.sub(r"he's", "he is", data)
    data = re.sub(r"there's", "there is", data)
    data = re.sub(r"We're", "We are", data)
    data = re.sub(r"That's", "That is", data)
    data = re.sub(r"won't", "will not", data)
    data = re.sub(r"they're", "they are", data)
    data = re.sub(r"Can't", "Cannot", data)
    data = re.sub(r"wasn't", "was not", data)
    data = re.sub(r"don\x89\u00b0t", "do not", data)
    data = re.sub(r"aren't", "are not", data)
    data = re.sub(r"isn't", "is not", data)
    data = re.sub(r"What's", "What is", data)
    data = re.sub(r"haven't", "have not", data)
    data = re.sub(r"hasn't", "has not", data)
    data = re.sub(r"There's", "There is", data)
    data = re.sub(r"He's", "He is", data)
    data = re.sub(r"It's", "It is", data)
    data = re.sub(r"You're", "You are", data)
    data = re.sub(r"I'M", "I am", data)
    data = re.sub(r"shouldn't", "should not", data)
    data = re.sub(r"wouldn't", "would not", data)
    data = re.sub(r"i'm", "I am", data)
    data = re.sub(r"I\x89\u00b0m", "I am", data)
    data = re.sub(r"I'm", "I am", data)
    data = re.sub(r"Isn't", "is not", data)
    data = re.sub(r"Here's", "Here is", data)
    data = re.sub(r"you've", "you have", data)
    data = re.sub(r"you\x89\u00b0ve", "you have", data)
    data = re.sub(r"we're", "we are", data)
    data = re.sub(r"what's", "what is", data)
    data = re.sub(r"couldn't", "could not", data)

```

```
data = re.sub(r"we've", "we have", data)
data = re.sub(r"it's", "it is", data)
data = re.sub(r"doesn't", "does not", data)
data = re.sub(r"It's", "It is", data)
data = re.sub(r"Here's", "Here is", data)
data = re.sub(r"who's", "who is", data)
data = re.sub(r"I've", "I have", data)
data = re.sub(r"y'all", "you all", data)
data = re.sub(r"can't", "cannot", data)
data = re.sub(r"would've", "would have", data)
data = re.sub(r"it'll", "it will", data)
data = re.sub(r"we'll", "we will", data)
data = re.sub(r"wouldn't", "would not", data)
data = re.sub(r"We've", "We have", data)
data = re.sub(r"he'll", "he will", data)
data = re.sub(r"Y'all", "You all", data)
data = re.sub(r"Weren't", "Were not", data)
data = re.sub(r"Didn't", "Did not", data)
data = re.sub(r"they'll", "they will", data)
data = re.sub(r"they'd", "they would", data)
data = re.sub(r"DON'T", "DO NOT", data)
data = re.sub(r"That's", "That is", data)
data = re.sub(r"they've", "they have", data)
data = re.sub(r"i'd", "I would", data)
data = re.sub(r"should've", "should have", data)
data = re.sub(r"You're", "You are", data)
data = re.sub(r"where's", "where is", data)
data = re.sub(r"Don't", "Do not", data)
data = re.sub(r"we'd", "we would", data)
data = re.sub(r"i'll", "I will", data)
data = re.sub(r"weren't", "were not", data)
data = re.sub(r"They're", "They are", data)
data = re.sub(r"Can't", "Cannot", data)
data = re.sub(r"you'll", "you will", data)
data = re.sub(r"I'd", "I would", data)
data = re.sub(r"let's", "let us", data)
data = re.sub(r"it's", "it is", data)
data = re.sub(r"can't", "cannot", data)
data = re.sub(r"don't", "do not", data)
data = re.sub(r"you're", "you are", data)
data = re.sub(r"i've", "I have", data)
data = re.sub(r"that's", "that is", data)
data = re.sub(r"i'll", "I will", data)
```

```

data = re.sub(r"doesn't", "does not",data)
data = re.sub(r"i'd", "I would", data)
data = re.sub(r"didn't", "did not", data)
data = re.sub(r"ain't", "am not", data)
data = re.sub(r"you'll", "you will", data)
data = re.sub(r"I've", "I have", data)
data = re.sub(r"Don't", "do not", data)
data = re.sub(r"I'll", "I will", data)
data = re.sub(r"I'd", "I would", data)
data = re.sub(r"Let's", "Let us", data)
data = re.sub(r"you'd", "You would", data)
data = re.sub(r"It's", "It is", data)
data = re.sub(r"Ain't", "am not", data)
data = re.sub(r"Haven't", "Have not", data)
data = re.sub(r"Could've", "Could have", data)
data = re.sub(r"youve", "you have", data)
data = re.sub(r"don't", "do not", data)

return data

```

g) Remove Punctuations & others

```

#Removes Punctuations
def remove_punctuations(data):
    punct_tag=re.compile(r'^[\w\s]')
    data=punct_tag.sub(r'',data)
    return data

```

h) Lemmatization

Đầu tiên cài và unzip file wordnet của thư viện “nltk”:

```

# Downloading data
import nltk
import subprocess

# Download and unzip wordnet
try:
    nltk.data.find('wordnet.zip')
except:
    nltk.download('wordnet', download_dir='/kaggle/working/')
    command = "unzip /kaggle/working/corpora/wordnet.zip -d /kaggle/working/corpora"
    subprocess.run(command.split())

```

```
nltk.data.path.append('/kaggle/working/')

# Now you can import the NLTK resources as usual
from nltk.corpus import wordnet
```

Sau đó, import và download lại:

```
import nltk
from nltk.stem import WordNetLemmatizer
nltk.download('wordnet')

from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
from nltk import pos_tag

def lemmatize_text_pos(text):
    lemmatizer = WordNetLemmatizer()
    punctuations = "?!.,"

    words = word_tokenize(text)
    words = [word for word in words if word not in punctuations]

    lemmatized_text = []
    for word in words:
        lemma = lemmatizer.lemmatize(word, pos="v")
        lemmatized_text.append(lemma)

    return " ".join(lemmatized_text)
```

i) Final Lower casing

```
#Lower casing
def lowercase(text):
    return text.lower()
```

## 4.3. Cài đặt mô hình và notebook dành cho quá trình huấn luyện Binary Classification

### 4.3.1. Thiết lập môi trường tính toán và đảm bảo tính tái hiện

Đoạn mã đầu tiên trong quy trình là để thiết lập môi trường đảm bảo tính lặp lại và hiệu suất cho các tác vụ tính toán hoặc học máy.

Đầu tiên, import các thư viện cần thiết như “pandas” để xử lý dữ liệu, “numpy” để tính toán số học, và “torch” để làm việc với các mô hình học sâu.

Hàm “set\_seed” được định nghĩa để đặt seed ngẫu nhiên cho các thư viện này, đảm bảo kết quả có thể tái hiện được qua các lần chạy khác nhau. Sau đó, mã kiểm tra xem CUDA có khả dụng không để xác định thiết bị thực hiện tính toán: Nếu có, nó sử dụng GPU để tăng tốc độ, nếu không thì dùng CPU.

Cuối cùng, thiết bị được sử dụng sẽ được xuất ra, giúp xác định nơi mà các phép tính sẽ diễn ra, đảm bảo rằng mã có thể chạy nhất quán và hiệu quả.

```
import pandas as pd
import re, string
import numpy as np
import random, torch

def set_seed(seed):
    random.seed(seed)
    np.random.seed(seed)
    torch.manual_seed(seed)
    if torch.cuda.is_available():
        torch.cuda.manual_seed_all(seed)

set_seed(42)

device = "cuda:0" if torch.cuda.is_available() else "cpu"
device
```

Việc thiết lập môi trường này rất quan trọng trong các dự án khoa học dữ liệu và học máy để đảm bảo sự ổn định và hiệu quả trong quá trình phát triển và triển khai mô hình.

#### 4.3.2. Import Train and Validation datasets

Khai báo đường dẫn đến các bộ dữ liệu, tại em sử dụng kaggle để nghiên cứu, nên đường dẫn là từ “/kaggle/input”:

```
path_train = "/kaggle/input/hope-task-2-final-
combined/final_train_combined.csv"
path_val = "/kaggle/input/hope-task-2-english/val_en.csv"
path_test = "/kaggle/input/hope-task-2-english/test_en.csv"
```

Sau đó đọc dữ liệu, tiền xử lý và chuyển đổi nhãn văn bản thành nhãn nhị phân để chuẩn bị dữ liệu cho các quá trình huấn luyện, fine-tune và đánh giá mô hình:

```
import pandas as pd

def convert_label(text):
    if text == "Hope":
        return 1
    else:
        return 0

def read_and_preprocessing(path_data):
    df = pd.read_csv(path_data)
    df["binary"] = df["binary"].apply(convert_label)
    x_input = df["text"].apply(preprocessing_text).tolist()
    y_output = df["binary"].tolist()
    ids = df["id"].tolist()
    return x_input, y_output, ids

train_texts, train_labels, train_ids =
read_and_preprocessing(path_train)
valid_texts, valid_labels, valid_ids =
read_and_preprocessing(path_val)
print(len(train_texts), len(train_labels))
print(len(valid_texts), len(valid_labels))

df_train = pd.DataFrame(list(zip(train_texts, train_labels)),
                        columns=['x_data', 'y_output'])
df_train.head()
```

**Tập dữ liệu train** sẽ là dữ liệu dùng cho việc huấn luyện mô hình.

**Tập dữ liệu validation** sẽ là dữ liệu dùng cho việc fine tune mô hình của chúng em để tìm ra tham số tốt nhất cho mô hình được cài đặt sẵn.

#### 4.3.3. Thiết lập mô hình BERT

**Import (Nhập) thư viện:**

- **‘AutoModelForSequenceClassification’**: Một lớp (class) từ thư viện Transformers của HuggingFace để tải mô hình được thiết kế cho nhiệm vụ phân loại chuỗi

- **‘AutoTokenizer’**: Một lớp để tải và sử dụng tokenizer từ thư viện Transformers của Hugging Face

```
import torch
from transformers import AutoModelForSequenceClassification,
AutoTokenizer
```

**Tải mô hình và Tokenizer:**

```
model_name = "FacebookAI/xlm-roberta-base"

bert_model =
AutoModelForSequenceClassification.from_pretrained(model_name,
ignore_mismatched_sizes=True)
tokenizer = AutoTokenizer.from_pretrained(model_name,
model_max_length=512, ignore_mismatched_sizes=True)
```

- Tải mô hình ‘xlm-roberta-base’ đã được huấn luyện trước, được thiết kế cho nhiệm vụ phân loại chuỗi.
- Tham số ‘ignore\_mismatched\_sizes=True’ cho phép bỏ qua sự khác biệt về kích thước giữa mô hình tải lên và mô hình mong đợi.
- **‘AutoTokenizer.from\_pretrained()’**: Tải tokenizer tương ứng với mô hình để mã hóa văn bản. Tham số ‘model\_max\_length=512’ đặt độ dài tối đa cho các chuỗi mã hóa là 512 token.

**Mã hóa văn bản huấn luyện:**

```
max_length = 512
train_encodings = tokenizer(train_texts, truncation=True,
max_length=max_length, padding=True)
```

- **“train\_encodings”**: mã hóa các văn bản dùng cho huấn luyện mô hình (train\_text). Tham số **‘truncation=True’** cắt bớt các văn bản dài hơn độ dài tối đa (512 token), và **‘padding=True’** đệm các văn bản ngắn hơn để đảm bảo tất cả các chuỗi đều có độ dài bằng 512 token.

**Định nghĩa hàm CustomDataset:** giúp tổ chức dữ liệu huấn luyện thành một định dạng mà mô hình học sâu có thể xử lý trực tiếp, đồng thời cho phép quản lý dữ liệu một cách linh hoạt và hiệu quả, theo đúng cách mà chúng ta mong muốn.

```
class CustomDataset(torch.utils.data.Dataset):
    def __init__(self, encodings, labels):
        self.encodings = encodings
```



```

self.labels = labels

def __getitem__(self, idx):
    item = {key: torch.tensor(val[idx]) for key, val in
self.encodings.items()}
    item['labels'] = torch.tensor(self.labels[idx])
    return item

def __len__(self):
    return len(self.labels)

```

- Lớp kế thừa từ 'torch.utils.data.Dataset'
- ' \_\_init\_\_ ' là phương thức khởi tạo của lớp  
'encodings' là một dictionary chứa các dữ liệu đã được mã hóa, ở đây là các token từ văn bản  
'label' là một danh sách chứa các nhãn (labels) tương ứng với phần tử/sample trong 'encodings'
- ' \_\_getitem\_\_ ' là phương thức trả về một phần tử từ dataset tại vị trí 'idx'  
Nó tạo ra một dictionary('item') chứa các tensor của PyTorch cho các giá trị đã mã hóa ('encodings') tại chỉ số 'idx'.  
Nó cũng thêm nhãn tương ứng ('labels') vào dictionary này.  
Phương thức này đảm bảo rằng dữ liệu được truy cập theo cách nhất quán và sẵn sàng để đưa vào mô hình học sâu
- ' \_\_len\_\_ ' trả về số lượng phần tử trong dataset  
Điều này cho phép PyTorch biết được độ dài của dataset, điều cần thiết khi sử dụng 'DataLoader' để tạo các lô dữ liệu.

### Thiết kế và bắt đầu quá trình huấn luyện:

```


from transformers import Trainer, TrainingArguments

training_args = TrainingArguments(
    output_dir='./results',          # output directory
    num_train_epochs=10,             # total number of training epochs
    learning_rate=2e-5,              # learning rate
    per_device_train_batch_size=32,  # batch size per device during
training
    warmup_steps=100,                # number of warmup steps for
learning rate scheduler
    weight_decay=0.01,               # strength of weight decay
    logging_dir='./logs',            # directory for storing logs

```

```

logging_steps=100,
save_total_limit = 1,
report_to="tensorboard"
)

trainer = Trainer(
    model=bert_model,                # the instantiated 
    Transformers model to be trained
    args=training_args,              # training arguments, defined above
    train_dataset=train_dataset,      # training dataset
)
trainer.train()

```

- ‘Trainer’ và ‘TrainingArguments’ là 2 lớp trong thư viện ‘Transformers’.  
‘Trainer’ giúp đơn giản hóa việc huấn luyện mô hình.  
‘TrainingArguments’ là một lớp chứa các tham số và cấu hình cho quá trình huấn luyện.
- **Lựa chọn các tham số cho mô hình huấn luyện:**  
**output\_dir=’./results’:** đường dẫn tới thư mục lưu trữ các kết quả huấn luyện.  
**num\_train\_epochs=10:** là tổng số epoch để huấn luyện mô hình, mỗi epoch là một lần duyệt qua toàn bộ dataset, ở đây là 10.  
**learning\_rate=2e-5:** tham số này điều chỉnh bước nhảy khi cập nhật trọng số của mô hình.  
**per\_device\_train\_batch\_size=32:** điều chỉnh kích thước của batch trong quá trình huấn luyện cho mỗi thiết bị.  
**warmup\_steps=100:** là số bước warmup (khởi động) cho việc điều chỉnh learning rate, giúp tránh việc thay đổi learning rate quá nhanh ngay từ đầu.  
**weight\_decay=0.01:** mức độ weight\_decay, giúp ngăn ngừa overfitting bằng cách phạt các trọng số lớn.  
**logging\_dir=’./logs’:** đường dẫn tới thư mục lưu trữ các log của quá trình huấn luyện.  
**logging\_steps=100:** số bước giữa mỗi lần ghi log.  
**save\_total\_limit=1:** giới hạn số lượng checkpoint được lưu giữ, chỉ lưu checkpoint mới nhất.  
**report\_to=’tensorboard’:** báo cáo kết quả huấn luyện lên TensorBoard, một công cụ trong thư viện để theo dõi và trực quan hóa quá trình huấn luyện, có thể xóa và lấy API để theo dõi kết quả

huấn luyện để điều chỉnh tham số (fine-tuning).

- **Khởi tạo ‘trainer’:**

**model=bert\_model:** các mô hình được tải trước sẽ được gọi để huấn luyện, ở đây là mô hình ‘xlm-roberta-base’.

**args = training\_args:** các tham số và cấu hình huấn luyện đã được định nghĩa trước sẽ được sử dụng.

**train\_dataset=train\_dataset:** dataset huấn luyện lấy ra khi gọi từ lớp ‘CustomDataset’ chứa các văn bản và nhãn đã được mã hóa.

**trainer.train():** Bắt đầu quá trình huấn luyện. ‘Trainer’ sẽ tự động quản lý quá trình huấn luyện, bao gồm việc cập nhật trọng số, ghi log và lưu checkpoint: Lấy dữ liệu và chia thành các batch dựa trên kích thước cung cấp (32). Trong mỗi batch, mô hình sẽ dự đoán nhãn cho các văn bản, so sánh với nhãn thực tế để tính toán lỗi, và sau đó cập nhật trọng số dựa trên gradient descent. Các thông tin về quá trình huấn luyện (giá trị lỗi, chỉ số hiệu suất) sẽ được ghi lại và xem trên TensorBoard. Trainer cũng sẽ lưu checkpoint của mô hình sau mỗi epoch hoặc theo lịch trình cụ thể.

#### 4.3.4. Inference + Evaluation

Sử dụng mô hình đã được huấn luyện để dự đoán nhãn cho các văn bản đánh giá, sau đó đánh giá hiệu suất của mô hình bằng cách so sánh các nhãn dự đoán với các nhãn thực tế, và tính toán các chỉ số như Accuracy, Precision, Recall và F1-Score.

```
def make_prediction(review,tokenizer,trainer):  
    demo_input = preprocessing_text(review)  
    demo_encodings = tokenizer([demo_input], truncation=True,  
max_length = max_length, padding=True)  
    test_dataset = CustomDataset(demo_encodings, [0])  
    predic_demo = trainer.predict(test_dataset)[0]  
    predict_label = np.argmax(predic_demo, axis=1).flatten().tolist()[0]  
    return predict_label  
  
y_pred = []  
  
for review in valid_texts:  
    a = make_prediction(review,tokenizer,trainer)  
    y_pred.append(a)
```

```

print(y_pred[:10])
print(valid_labels[:10])

from sklearn.metrics import *

print("M_Pr: ", round(precision_score(valid_labels, y_pred,
average='macro'),4))
print("M_Re: ", round(recall_score(valid_labels, y_pred,
average='macro'),4))
print("M_F1: ", round(f1_score(valid_labels, y_pred,
average='macro'),4))

print("W_Pr: ", round(precision_score(valid_labels, y_pred,
average='weighted'),4))
print("W_Re: ", round(recall_score(valid_labels, y_pred,
average='weighted'),4))
print("W_F1: ", round(f1_score(valid_labels, y_pred,
average='weighted'),4))

print("acc:", round(accuracy_score(valid_labels, y_pred), 4))

```

- Hàm **‘make\_prediction’**: nhận vào, mã hóa, và đánh giá một văn bản đầu vào:  
**‘demo\_input = preprocessing\_text(review)’**: tiền xử lý văn bản cần đánh giá  
**‘demo\_encodings’**: mã hóa văn bản với tokenizer, cắt bớt (truncation) và đệm (padding) để đảm bảo độ dài  
**‘test\_dataset’** : tạo dataset từ văn bản đã mã hóa, sử dụng lớp **‘CustomDataset’** đã định nghĩa  
**‘predict\_demo’**: dự đoán nhãn cho dataset, kết quả trả về là xác suất cho các lớp.  
**‘predict\_label’**: lấy nhãn có xác suất cao nhất bằng cách sử dụng hàm **‘np.argmax’**  
**Cuối cùng trả về nhãn dự đoán.**
- Sau đó sử dụng hàm **‘make\_prediction’** để dự đoán nhãn cho mỗi văn bản trong tập dữ liệu đánh giá (**‘valid\_texts’**) và lưu các nhãn dự đoán vào danh sách **‘y\_pred’**.
- Cuối cùng là đánh giá hiệu suất mô hình:  
 Tính toán các chỉ số đánh giá hiệu suất của mô hình, sử dụng **‘sklearn.metrics’** để dùng những hàm tính toán chỉ số.

Đoạn mã thực hiện một quy trình hoàn chỉnh để dự đoán và đánh giá hiệu suất của mô hình học sâu đã được huấn luyện trên tập dữ liệu validation. Nó sử dụng mô hình để dự đoán cho các văn bản trong tập validation, sau đó so sánh nhãn dự đoán với nhãn thực tế và tính toán các chỉ số đánh giá để xác định hiệu suất của mô hình. Mục tiêu của phần này chính là sử dụng tập Validation để hiểu rõ mức độ chính xác và độ hiệu quả của mô hình, từ đó điều chỉnh các tham số để tìm ra bộ tham số cho ra kết quả tốt nhất. Trong quá trình nghiên cứu đề tài, tụi em sử dụng phần code này nhiều lần để tìm ra cách cài đặt tham số tốt nhất cho mỗi mô hình sử dụng trước khi đưa vào đánh giá trên tập kiểm thử.

#### 4.3.5. Đánh giá trên tập kiểm thử

Đọc dữ liệu và tiền xử lý dữ liệu riêng cho tập test, sau đó dùng hàm **'make\_prediction'** để gán nhãn cho các mẫu dữ liệu trong tập kiểm thử:

```
test_texts, test_labels, test_ids = read_and_preprocessing(path_test)

y_pred_test = []

for review in test_texts:
    a = make_prediction(review, tokenizer, trainer)
    y_pred.append(a)
```

Sử dụng thư viện “sklearn.metrics” với các hàm tính toán chỉ số để tính toán các độ đo đánh giá hiệu quả của mô hình:

```
from sklearn.metrics import *

print("M_Pr: ", round(precision_score(test_labels, y_pred_test,
average='macro'),4))
print("M_Re: ", round(recall_score(test_labels, y_pred_test,
average='macro'),4))
print("M_F1: ", round(f1_score(test_labels, y_pred_test,
average='macro'),4))

print("W_Pr: ", round(precision_score(test_labels, y_pred_test,
average='weighted'),4))
print("W_Re: ", round(recall_score(test_labels, y_pred_test,
average='weighted'),4))
```

```
print("W_F1: ", round(f1_score(test_labels, y_pred_test,
average='weighted'),4))

print("acc:", round(accuracy_score(test_labels, y_pred_test), 4))
```

## 4.4. Cài đặt mô hình và notebook dành cho quá trình huấn luyện Multiclass Classification

Nhìn chung, đối với phần Multiclass Classification trong đề tài, cài đặt mô hình từ đầu đến cuối không khác gì so với cài đặt cho phần Binary Classification, chỉ khác chỗ encoding nhãn cho văn bản, nhưng tui em quyết định thiết kế riêng để dễ theo dõi và đánh giá. Cụ thể khác phần sau:

### Hàm chuyển đổi nhãn từ dạng văn bản sang dạng số

```
def convert_label(text):
    if text == "Not Hope":
        return 0
    elif text == "Generalized Hope":
        return 1
    elif text == "Unrealistic Hope":
        return 2
    elif text == "Realistic Hope":
        return 3
    else:
        print("Error: ", text)
        return 0
```

- Hàm này xử lý nhiều loại nhãn khác nhau (tổng cộng bốn loại nhãn) cho bài toán phân loại đa lớp

## 4.5. Kết quả thử nghiệm

### 4.5.1. Kết quả mô hình Binary Classification trên tập kiểm thử:

Bởi vì tui em thử nghiệm trên khá nhiều mô hình Transformers BERT để so sánh, nên tui em có làm bảng sau để mô tả tổng quát kết quả thử

nghiệm:

Experimental result of Binary Classification

Datasets	Models	Binary Classification Result						
		M_Pr	M_Re	M_F1	W_Pr	W_Re	W_F1	Acc
Simple Preprocessing	<b>XLM-R-base</b>	<b>86.65%</b>	<b>86.53%</b>	<b>86.58%</b>	<b>86.64%</b>	<b>86.63%</b>	<b>86.62%</b>	<b>86.63%</b>
	DeBERTa-v3-base	85.62%	85.15%	85.26%	85.52%	85.37%	85.32%	85.37%
	Twitter-XLM-roBERTa	85.34%	84.59%	84.73%	85.20%	84.88%	84.80%	84.88%
Enhanced Preprocessing	XLM-R-base	86.08%	85.94%	85.99%	86.06%	86.05%	86.03%	86.05%
Data Augmentation	XLM-R-base	83.25%	83.32%	83.23%	83.37%	83.24%	83.25%	83.25%

Nhìn chung, bảng “Experimental result of Binary Classification” cho thấy hiệu suất của các mô hình khác nhau trên các bộ dữ liệu khác nhau. Cụ thể:

- Mô hình “XLM-R-base” có hiệu suất, kết quả tốt nhất trên bộ dữ liệu “Simple Preprocessing” với Accuracy là 86,65%.
- Mô hình “DeBERTa-v3-base” cho ra kết quả tương đối với độ chính xác (Accuracy - Acc) là 85,37%.
- Mô hình “Twitter-XLM-roBERTa” với thể mạnh trong các bài toán liên quan đến dữ liệu mạng xã hội lại cho ra kết quả mặc dù thấp nhất nhưng vẫn khá cao với Accuracy là 84,88%.

Sau khi thử 3 mô hình trên tập dữ liệu chỉ được xử lý đơn giản và tìm được mô hình cho ra kết quả tốt nhất là XLM-R-base, tụi em sử dụng mô hình này kết hợp với các phương pháp tiền xử lý dữ liệu khác là “Tiền xử lý dữ liệu phức tạp hơn – Enhanced Preprocessing” và trên bộ dữ liệu đã áp dụng phương pháp “Data Augmentation”. Kết quả là cả 2 phương pháp đều cho ra kết quả không khả quan hơn ban đầu:

- Trên bộ dữ liệu được tiền xử lý dữ liệu phức tạp hơn cho ra độ chính xác (Accuracy) là 86.05 %.
- Đối với kết quả sau khi áp dụng phương pháp Data Augmentation, Accuracy đạt được là 83,25%, thấp hơn 3,4% so với ban đầu.

#### 4.5.2. Kết quả mô hình Multiclass Classification trên tập kiểm thử:

Tương tự, dưới đây là kết quả tổng quát quá trình thử nghiệm của nhóm tụi em đối với mô hình Multiclass Classification:

Experimental result of Multiclass Classification

Datasets	Model	Multiclass Classification						
		M_Pr	M_Re	M_F1	W_Pr	W_Re	W_F1	Acc
Simple Preprocess	<b>DeBERTa-v3-base</b>	<b>69.19%</b>	<b>70.80%</b>	<b>69.92%</b>	<b>76.33%</b>	<b>75.58%</b>	<b>75.89%</b>	<b>75.58%</b>
	Twitter-XLM-roBERTa	68.27%	69.85%	69.00%	75.63%	75.10%	75.32%	75.10%
Enhanced Preprocess	Twitter-XLM-roBERTa	68.60%	70.21%	69.34%	76.03%	75.39%	75.65%	75.39%
Data Augmentaion	Twitter-XLM-roBERTa	66.31%	66.11%	66.16%	72.14%	72.38%	72.21%	72.38%

Nhìn chung, trên bộ dữ liệu chỉ được tiền xử lý dữ liệu đơn giản, các mô hình lần lượt cho ra kết quả:

- Mô hình “DeBERTa-v3-base” cho ra kết quả với độ chính xác là 75,58%, cao nhất trong các mô hình thử nghiệm.
- Mô hình “Twitter-XLM-roBERTa cho ra độ chính xác khi dự đoán trên tập kiểm thử là 75,10%.

Lần này, tụi em quyết định thử nghiệm các biện pháp tiền xử lý dữ liệu khác trên mô hình “Twitter-XLM-roBERTa” chứ không phải là mô hình có kết quả cao nhất là “DeBERTa-v3-base”. Bởi vì mục đích tụi em sử dụng mô hình lần này là vì đây là mô hình được huấn luyện trước (pretrained) dành cho bài toán với dữ liệu là mạng xã hội. Nên nhằm mục đích thử nghiệm các phương pháp tiền xử lý của tụi em có tác dụng trên các mô hình này không thì tụi em quyết định thử nghiệm sâu hơn với mô hình này. Kết quả là:

- Trên bộ dữ liệu được tiền xử lý phức tạp hơn, mô hình cho ra kết quả với độ chính xác cao hơn 0,3%.
- Về mặt kết quả sau khi áp dụng phương pháp Data Augmentation, mô hình cho ra kết quả trên tập thử nghiệm có độ chính xác là 72,38%, không khả quan hơn ban đầu.

## 4.6. Mô hình SVM sử dụng GridSearch và TfidfVectorizer ở mức độ cơ bản:

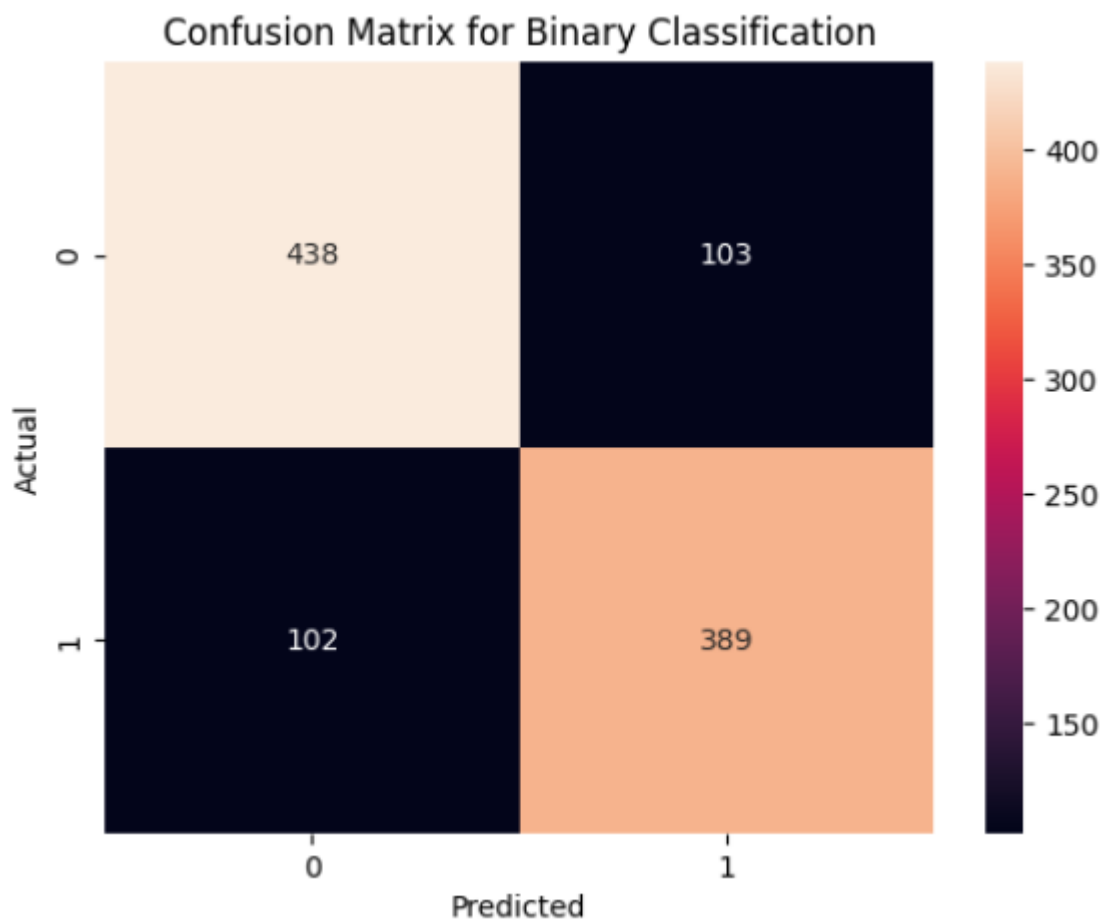
### 4.6.1. Kết quả phần Binary Classification:

**Classification report:**



	precision	recall	f1-score	support
Hope	0.81	0.81	0.81	541
Not Hope	0.79	0.79	0.79	491
accuracy			0.80	1032
macro avg	0.80	0.80	0.80	1032
weighted avg	0.80	0.80	0.80	1032

#### Confusion matrix:

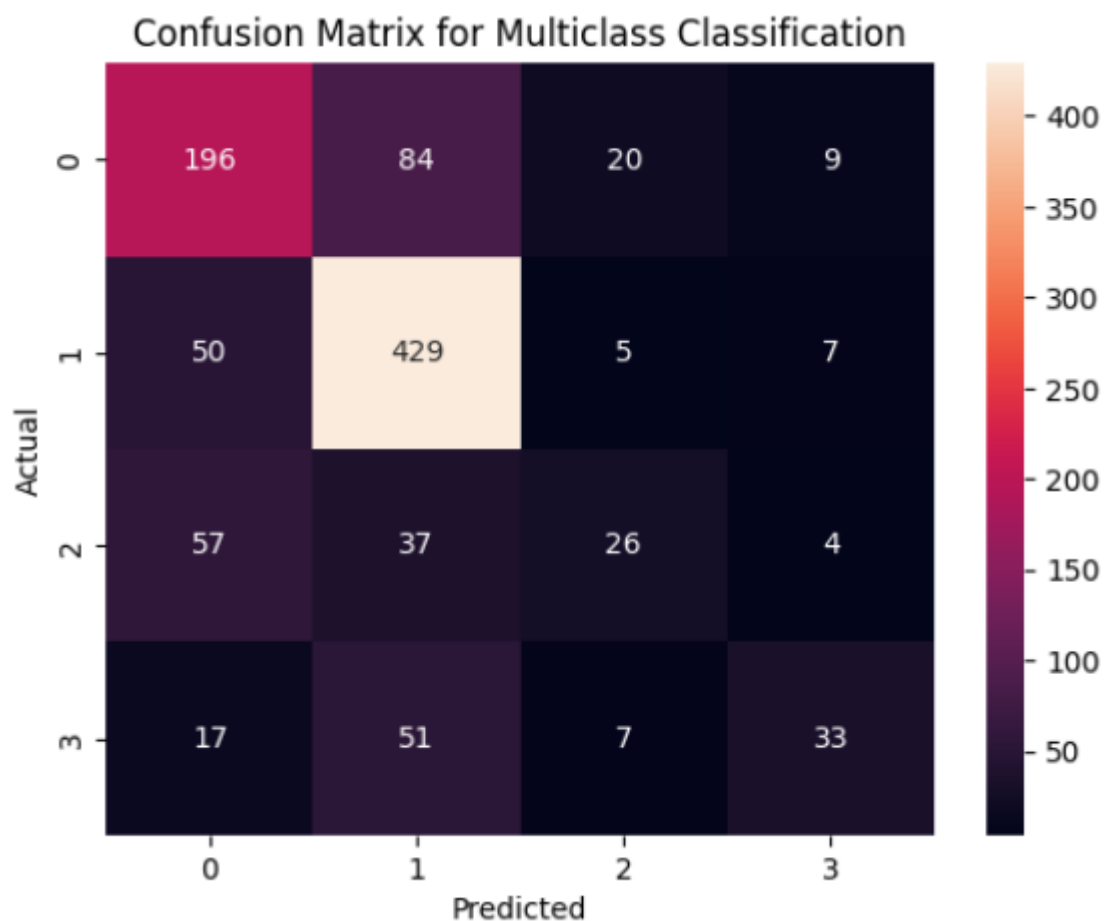


#### 4.6.2. Kết quả phân Multiclass Classification:

##### Classification report:

	precision	recall	f1-score	support
Generalized Hope	0.61	0.63	0.62	309
Not Hope	0.71	0.87	0.79	491
Realistic Hope	0.45	0.21	0.29	124
Unrealistic Hope	0.62	0.31	0.41	108
accuracy			0.66	1032
macro avg	0.60	0.51	0.53	1032
weighted avg	0.64	0.66	0.64	1032

**Confusion matrix:**



#### 4.6.3. Tổng kết

Nhìn chung, mô hình SVM được cài đặt sử dụng các tham số tốt nhất tìm được bằng GridSearchCV và có sử dụng phương pháp trích xuất đặc trưng là Tfidf (sử dụng thư viện). Song cho ra kết quả không khả quan hơn so với phương pháp chính của tụi em. Cụ thể như sau:

- **Binary Classification Result:** mô hình dự đoán trên tập test với Accuracy đạt được là 80%, thấp hơn 6,63% so với kết quả tốt nhất từ các mô hình Transformers BERT.
- **Multiclass Classification Result:** mô hình dự đoán trên tập test với Accuracy đạt được là 66%, thấp hơn gần 10% so với kết quả tốt nhất từng đạt được trước đó khi sử dụng các mô hình Transformers BERT.

## Chương 5. Kết luận

Trong nghiên cứu này, chúng em đã thực hiện việc phát hiện hy vọng trong các văn bản từ mạng xã hội bằng cách sử dụng các mô hình ngôn ngữ tiên tiến dựa trên thư viện Transformers từ Hugging Face kết hợp phương pháp tăng cường dữ liệu (Data Augmentation). Kết quả từ việc áp dụng các mô hình ngôn ngữ cùng những bước tiền xử lý khác nhau và phương pháp Data Augmentation đã cho ra kết quả khả quan trong việc phân loại văn bản có chứa Hy vọng và Không chứa hy vọng. Cụ thể, chúng em đã đạt được những kết quả tích cực đối với cả 2 nhiệm vụ phân loại nhị phân (Binary Classification) và phân loại đa lớp (Multiclass Classification), cho thấy khả năng của các mô hình này trong việc nắm bắt và hiểu rõ cảm xúc của các bài đăng trên mạng xã hội.

### 5.1. Các điểm nổi bật rút ra từ quá trình thực hiện đề tài:

- **Hiệu quả của các mô hình BERT:** Việc sử dụng các mô hình BERT và các biến thể của nó đã giúp chúng em đạt được hiệu suất cao trong việc phân loại cảm xúc hy vọng từ dữ liệu văn bản. Các mô hình này không chỉ thể hiện khả năng vượt trội so với các mô hình truyền thống mà còn cung cấp độ chính xác cao hơn trong việc xử lý ngôn ngữ tự nhiên.
- **Tầm quan trọng của tiền xử lý dữ liệu:** Các bước tiền xử lý dữ liệu đã chứng minh vai trò quan trọng trong việc cải thiện hiệu suất của mô hình. Việc loại bỏ các yếu tố nhiễu như liên kết URL, các ký tự đặc biệt, và chuyển đổi các từ viết tắt thành phiên bản đầy đủ cấu trúc,... Đã giúp mô hình hiểu rõ hơn về ý nghĩa của văn bản
- **Ứng dụng của Data Augmentation (Tăng cường dữ liệu):** Các kỹ thuật tăng cường dữ liệu đã giúp mở rộng và làm phong phú dữ

liệu huấn luyện, từ đó giúp mô hình mô hình học hỏi từ nhiều tình huống và ngữ cảnh khác nhau, nâng cao khả năng tổng quát hóa của mô hình.

## **5.2. Đóng góp của đề tài**

Nghiên cứu trong đề tài này đã cung cấp một phương pháp tiếp cận hiệu quả trong việc phát hiện cảm xúc hy vọng từ dữ liệu văn bản mạng xã hội. Kết quả của nghiên cứu không chỉ có ý nghĩa đối với việc phân tích cảm xúc trong các nghiên cứu học thuật mà còn có thể được ứng dụng trong các hệ thống phân tích cảm xúc thực tế, giúp cải thiện khả năng tương tác và hỗ trợ người dùng trên các nền tảng truyền thông xã hội.

## **5.3. Hạn chế và hướng phát triển**

Mặc dù đạt được những kết quả đáng kể, nghiên cứu của tui em cũng gặp một số hạn chế:

- Việc thiết kế các phương pháp tiền xử lý dữ liệu vẫn chưa được tối ưu, chỉ cải thiện một cách khách quan đối với vài mô hình đã áp dụng.
- Trong quá trình thiết kế phương pháp Data Augmentation – Tăng cường dữ liệu sử dụng Large Language Model (Tui em sử dụng Gemini), tui em thiết kế các prompt vẫn chưa hợp lý, nên vẫn chưa cải thiện được kết quả nghiên cứu. Một phần có thể vì sử dụng các mô hình Transformers BERT đã cho kết quả khá tốt nên dẫn đến việc khó cải thiện.

Trong tương lai, tui em sẽ tập trung vào việc cải tiến các mô hình và các phương pháp tiền xử lý để nâng cao hiệu suất của hệ thống. Đồng thời, việc khám phá các kỹ thuật học sâu mới và ứng dụng vào các bài toán Sentiment Analysis cũng là hướng đi tiềm năng để phát triển nghiên cứu đề tài này.

## **5.4. Tổng kết.**

Kết quả từ nghiên cứu trong đề tài này của tui em đã chứng minh rằng việc kết hợp các mô hình ngôn ngữ tiên tiến với các kỹ thuật tiền xử lý và Data Augmentation (Tăng cường dữ liệu) có thể tạo ra một hệ thống phát hiện cảm xúc hy vọng hiệu quả, góp phần vào việc phân tích và hiểu rõ hơn về cảm xúc của người dùng trên các nền tảng mạng xã hội.

## Các tài liệu tham khảo:

- [1] García-Baena, “Overview of HOPE at IberLEF 2024: Approaching Hope Speech Detection in Social Media from Two Perspectives, for Equality, Diversity and Inclusion and as Expectations,” 2024.
- [2] L. a. J.-Z. S. M. a. R. F. Chiruzzo, “Overview of IberLEF 2024: Natural Language Processing Challenges for Spanish and other Iberian **Languages**,” 2024.
- [3] D. Ezzy, “Illness narratives: Time, hope and HIV,” 2000.
- [4] R. a. C. G. a. H. S. a. C. V. Wiles, “The Management of Confidentiality and Anonymity in Social Research,” 2008.
- [5] S. Webb, “The Effects of Repetition on Vocabulary Knowledge,” 2007.
- [6] D. P. a. J. Šnajder, “Data Augmentation for Neural NLP,” 2023.
- [7] A. a. Conneau, “Unsupervised Cross-lingual Representation Learning at Scale,” Association for Computational Linguistics, 2020.
- [8] P. a. L. X. a. G. J. a. C. W. He, “DEBERTA: DECODING-ENHANCED BERT WITH DISENTANGLED ATTENTION,” 2020.
- [9] F. a. Barbieri, “{XLM}-{T}: Multilingual Language Models in {T}witter for Sentiment Analysis and Beyond,” European Language Resources Association, 2022.