Tuần 4 - Hệ thống phân loại message HAM và SPAM

Time-Series Team

Ngày 2 tháng 8 năm 2025

Dự án Hệ thống phân loại message HAM và SPAM bao gồm hai nội dung chính:

- ***Phần I: Mô hình phân loại tin nhắn Ham và Spam phiên bản mở rộng***
- ***Phần II: Streamlit***

Mục lục

I. Mô hình phân loại tin nhắm Ham và Spam phiên bản mở rộng

1. Mô tả bài toán

Trong thời đại số bùng nổ như hiện nay, mỗi người dùng Internet đều phải tiếp nhận hàng trăm, thậm chí hàng ngàn tin nhắn điện tử mỗi tuần. Tuy nhiên, không phải tin nhắn nào cũng đáng tin cậy. Bên cạnh những email hữu ích phục vụ công việc và đời sống (gọi là **ham**), ngày càng nhiều **email rác (spam)** xuất hiện với nội dung ngày càng *tinh vi*, *ẩn dụ*, và *khó phân biệt*.

Không chỉ dừng lại ở các quảng cáo thông thường, nhiều tin nhắn spam **giả mạo cú pháp của email hợp lệ**, sử dụng **ngôn từ tin cậy**, **đánh vào tâm lý người dùng** để lừa đảo, đánh cắp dữ liệu hoặc trục lợi tài chính. Điều này đặt ra thách thức lớn cho mô hình học máy, khi cần phân biệt **ranh giới mờ nhạt giữa spam và ham**, đặc biệt trong bối cảnh mà **email hợp lệ đôi khi cũng chứa những từ khóa nhạy cảm dễ bị nhầm lẫn với spam**.

Email Spam là gì?

Email spam là những email **không mong muốn**, thường được gửi hàng loạt với mục đích quảng cáo, lừa đảo, hoặc lan truyền nội dung độc hại. Người dùng **không chủ động đăng ký** nhận những email này.

Ví dụ (Spam Email):

- *"Win a brand new iPhone today! Just click this link to claim!"*
- *"You've been selected for a $1000 Walmart gift card!"*
- *"Invest in crypto now and double your money overnight!"*

Dấu hiệu nhận biết:

- Có từ khóa gây kích thích: `free, urgent, claim now, winner...`
- Chứa link rút gọn, domain lạ
- Có nhiều lỗi chính tả, viết hoa bất thường
- Nội dung hời hợt, phi logic

Email Ham là gì?

Email ham là những email **hợp lệ**, được gửi từ **nguồn tin cậy** như bạn bè, đồng nghiệp, trường học, ngân hàng,... và thường có nội dung cụ thể, liên quan trực tiếp đến người nhận.

Ví dụ (Ham Email):

- *"Hi John, just a reminder that your doctor's appointment is at 3PM today."*
- *"Your monthly salary has been transferred to your account."*
- *"Please review the attached report before the meeting tomorrow."*

Dấu hiệu nhận biết:

- Gửi từ người hoặc tổ chức đã biết
- Ngữ điệu rõ ràng, cụ thể
- Không có từ khóa dụ dỗ hay bất thường
- Nội dung liên quan trực tiếp đến cá nhân người nhận

Các lĩnh vực dễ bị nhầm lẫn giữa spam và ham

Trong những năm gần đây, sự phát triển của công nghệ email marketing và các hình thức lừa đảo trực tuyến đã dẫn đến sự gia tăng mạnh mẽ của các loại **spam tinh vi** – những tin nhắn rác được *thiết kế cẩn thận để vượt qua các bộ lọc tự động*. Chúng thường sử dụng ngôn ngữ lịch sự, cú pháp tự nhiên như email thật, thậm chí mô phỏng cách viết của email công việc hoặc cá nhân.
Cùng lúc đó, cũng tồn tại nhiều email hợp lệ (**ham**) có chứa các từ khóa như *"transfer"*, *"discount"*, *"verify"* vốn thường xuất hiện trong spam, khiến hệ thống nhầm lẫn. Những trường hợp như vậy được gọi là **hard ham** – tức là các email hợp pháp nhưng có đặc điểm giống với spam.

Sự mờ ranh giới giữa ham và spam xuất phát từ hai phía:

- **Ham có nội dung "giống spam"**: Các email hợp lệ chứa từ ngữ nhạy cảm hoặc có cấu trúc giống các email quảng cáo, tài chính, cảnh báo.
- **Spam tinh vi "giả dạng ham"**: Các email spam được viết khéo léo như thư cảm ơn, thông báo bảo mật, yêu cầu hành động bình thường – khiến con người cũng có thể bị đánh lừa.

Vì vậy, các mô hình học máy nếu chỉ dựa vào keyword hoặc kỹ thuật phân loại đơn giản như TF-IDF, Naive Bayes,... sẽ khó đạt hiệu quả cao. Thay vào đó, mô hình cần có khả năng **hiểu sâu ngữ nghĩa**, kết hợp thông tin ngữ cảnh, cú pháp, và thậm chí cả lịch sử người gửi để đưa ra dự đoán chính xác.

Dưới đây là bảng tổng hợp các nhóm nội dung dễ gây nhầm lẫn – xuất hiện trong cả ham và spam tinh vi, đòi hỏi mô hình phải rất tinh tế mới phân biệt được:

Nhóm nội dung	Ví dụ nội dung	Dễ nhầm với
`financial_phrases`	"Please confirm the $200 transfer from your account." "Your invoice for June is now available."	Scam / Phishing
`promotion_phrases`	"Flash sale ends tonight – 30% off all items!" "Exclusive discount for HUST students."	Spam quảng cáo
`lottery_phrases`	"You've been selected for a loyalty reward." "You may be eligible for a lucky draw."	Spam quà tặng / Random Reward
`scam_alert_phrases`	"Unusual login detected. Was this you?" "A payment attempt was blocked on your card."	Cảnh báo giả / Giả danh ngân hàng
`call_to_action_phrases`	"Act now to secure your spot in the seminar." "Verify your email to complete registration."	Spam ép buộc / Confirmation bait

`social_engineering_phrases`	"Mom, I need $150 for the hospital bill." "Can you transfer me some money? It's urgent."	Lừa đảo cảm xúc / Human-based spam
`obfuscated_phrases`	"Cl!ck h3re f0r fr33 b0nus" "`Claim your gift now!!!`"	Spam mạo danh / Bypass filter

Các nhóm nội dung dễ gây nhầm lẫn giữa spam và ham

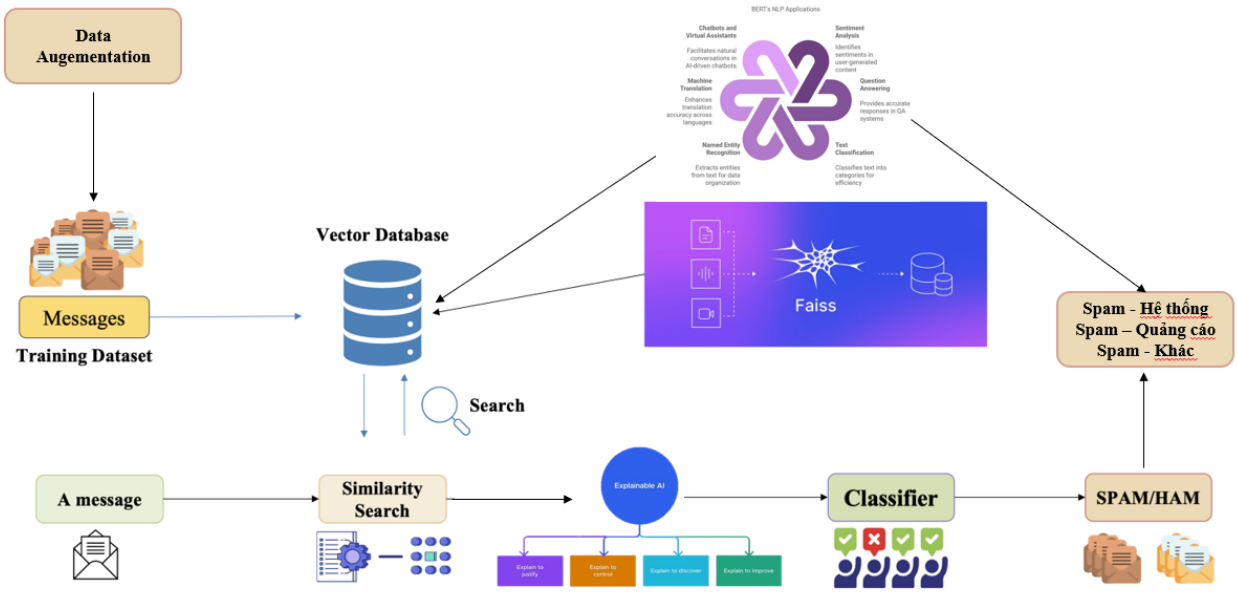
Các nội dung trên có thể hợp pháp nếu đến từ một nguồn uy tín (ví dụ: ngân hàng, hệ thống trường học, công ty thương mại điện tử). Tuy nhiên, nếu bị giả mạo hoặc viết bằng cách đánh lừa về mặt cảm xúc và hành vi, chúng sẽ trở thành spam nguy hiểm. Điều này đòi hỏi các mô hình phân loại hiện đại không chỉ dừng ở mức "biết keyword nào là xấu", mà phải **hiểu thông điệp tổng thể** và **phân tích toàn bộ cấu trúc thông tin trong email**.

Do đó, trong project này, mục tiêu nhóm đặt ra là xây dựng một mô hình học máy không chỉ phân loại chính xác giữa spam và ham, mà còn có khả năng **giải thích rõ ràng lý do dự đoán**, **mở rộng và đào sâu vào cải tiến kỹ thuật** giúp máy có thể ***hiểu ngữ nghĩa*** hơn. Điều này đặc biệt quan trọng trong các hệ thống chống spam hiện đại, giúp người dùng hiểu rõ liệu một email nên bị xóa, xem qua hay báo cáo. Ngoài ra, nhóm còn hướng đến việc **mở rộng phân loại chi tiết trong nhóm spam** (quảng cáo, hệ thống, lừa đảo, v.v...) nhằm tăng trải nghiệm và bảo mật cho người dùng.

Hệ thống phân loại tin nhắn spam/ham được thiết kế với cơ chế đầu vào – đầu ra như sau:

- **Input:** Một chuỗi văn bản đầu vào đại diện cho nội dung của một tin nhắn (dạng `string`). Tin nhắn này có thể là email, tin SMS, hoặc đoạn chat, bao gồm cả các trường hợp có cấu trúc bất thường như: viết tắt, lỗi chính tả, hoặc sử dụng ký tự đặc biệt
- **Output:** Một giá trị logic (dạng `bool`) phản ánh kết quả phân loại của hệ thống:
  - `True` nếu hệ thống xác định tin nhắn là **spam**.
  - `False` nếu tin nhắn được phân loại là **ham** (hợp lệ).

1.2. Nội dung bài toán



Hình 1: Enter Caption

Bài toán đặt ra là xây dựng một hệ thống phân loại tin nhắn thành **spam** hoặc **ham** không chỉ chính xác mà còn có khả năng giải thích được. Vượt trội hơn các phương pháp truyền thống, hệ thống này kết hợp sức mạnh của các mô hình ngôn ngữ lớn và kỹ thuật tìm kiếm tương đồng, tạo ra một giải pháp đáng tin cậy và minh bạch. Hệ thống đề xuất bao gồm các thành phần chính sau:

Data Augmentation:

- **Vấn đề tồn đọng:** Dữ liệu huấn luyện thường thiếu đa dạng và mất cân bằng nghiêm trọng giữa hai lớp *spam* và *ham*, làm giảm khả năng khái quát hóa của mô hình.
- **Giải pháp:** Sử dụng các kỹ thuật tăng cường dữ liệu để cải thiện độ đa dạng và cân bằng dữ liệu.
  - *Synonym Replacement:* Tạo các biến thể của tin nhắn bằng cách thay thế từ đồng nghĩa.
  - *Hard Ham Generation:* Tạo các tin nhắn *ham* khó có đặc điểm gần giống spam, buộc mô hình phải học các ranh giới phân loại tinh vi hơn.

Similarity Search (KNN-Classifier):

- **Vấn đề tồn đọng:** Phương pháp bỏ phiếu đa số (*majority vote*) đơn giản trong KNN bỏ qua mức độ quan trọng của từng hàng xóm.
- **Giải pháp:** Khi có một tin nhắn mới, hệ thống tìm kiếm những tin nhắn tương tự nhất. Quyết định phân loại được đưa ra bằng Weighted KNN, sử dụng độ tương đồng (*similarity score*) làm trọng số để ưu tiên các hàng xóm gần hơn.

Explainable AI (XAI) và Classifier:

- **Vấn đề tồn đọng:** Mô hình hoạt động như một "hộp đen", khó giải thích lý do đưa ra dự đoán. Khả năng giải thích thường bị tách rời khỏi quá trình phân loại chính.
- **Giải pháp:** Tích hợp khả năng giải thích vào lõi của bộ phân loại.
  - *Masking-based Saliency:* Phương pháp này xác định các từ khóa quan trọng nhất trong tin nhắn. Nói trực quan thì từ nào quan trọng trong quyết định spam hơn sẽ được tô đậm hơn.
  - *Phân loại có tích hợp Saliency:* Bộ phân loại sử dụng một tham số 'alpha' để điều chỉnh mức độ ảnh hưởng của điểm nổi bật (saliency score) vào công thức phân loại cuối cùng, giúp kết quả chính xác hơn và có thể giải thích được.

Đầu ra cuối cùng: Đầu ra cho mỗi câu gồm thông tin dự đoán và chỉ số giải thích cho dự đoán đó, giúp người dùng hiểu rõ quyết định của mô hình. Cấu trúc đầu ra bao gồm:

- **Lớp dự đoán:** Tin nhắn được gán nhãn dự đoán cuối cùng (*SPAM* hoặc *HAM*) dựa trên kết quả phân loại.
- **Saliency Weight:** Giá trị thể hiện mức độ ảnh hưởng của các từ khóa quan trọng trong tin nhắn đến quyết định phân loại cuối cùng. Nó được tính toán bằng cách kết hợp điểm nổi bật (*saliency score*). Tham số α là tham số điều chỉnh, quyết định mức độ ưu tiên của điểm saliency so với độ tương đồng tổng thể của tin nhắn.
- **Vote Scores:** Hệ thống hiển thị điểm số bỏ phiếu cho mỗi lớp (*Ham* và *Spam*). Dự đoán cuối cùng sẽ là lớp có điểm số cao nhất.
- **Spam Subcategory:** Nếu tin nhắn được phân loại là *SPAM*, hệ thống tiếp tục phân tích để xác định tiểu mục spam cụ thể (ví dụ: *spam_quangcao*, *spam_hethong*).
- **Cơ sở giải thích (Top neighbors):** Hệ thống liệt kê một số hàng xóm gần nhất trong cơ sở dữ liệu vector. Mỗi neighbors bao gồm:
  - *Nhãn (Label):* Nhãn của tin nhắn gốc (*ham* hoặc *spam*).
  - *Độ tương đồng (Similarity):* Giá trị thể hiện mức độ tương đồng giữa tin nhắn đầu vào và hàng xóm.
  - *Nội dung (Message):* Nội dung của tin nhắn hàng xóm.

1 Mô hình mở rộng

1.1 Data augmentation

1. Đặt vấn đề

Trong hệ thống phân loại tin nhắn spam/ham sử dụng mô hình embedding `multilingual-e5-base` kết hợp FAISS + kNN, pipeline ban đầu đã đạt được kết quả rất cao về độ chính xác:

- **Accuracy** đạt tới hơn 99%, đặc biệt với giá trị $k = 1$
- **Recall (spam)** dao động từ 95.3% đến 96.6% ứng với các giá trị $k = 1, 3, 5$

Trong hệ thống phân loại tin nhắn spam/ham, nhóm chúng mình triển khai pipeline kết hợp giữa mô hình embedding ngữ nghĩa `multilingual-e5-base` và thuật toán k-nearest neighbors (kNN) sử dụng chỉ mục FAISS để truy vấn các câu gần nhất. Mỗi tin nhắn đầu vào được ánh xạ thành vector embedding, sau đó được so sánh với tập huấn luyện bằng độ tương đồng cosine. Kết quả phân loại được đưa ra dựa trên biểu quyết đa số từ k hàng xóm gần nhất.

Khi tiến hành đánh giá trên tập kiểm tra, pipeline ban đầu cho thấy hiệu suất rất cao về độ chính xác:

- **Độ chính xác (Accuracy)** đạt tới hơn **99%** khi sử dụng $k = 1$, cho thấy rằng mô hình có khả năng khớp lại chính xác với các câu trong tập huấn luyện bằng embedding.
- **Recall đối với lớp `spam`** dao động từ **95.3% đến 96.6%** khi thay đổi giá trị k từ 1 đến 5, phản ánh năng lực phát hiện tin nhắn rác vẫn được duy trì ổn định ở nhiều thiết lập khác nhau.

Tuy nhiên, kết quả "độ chính xác rất cao với $k = 1$" có thể là dấu hiệu cho một số vấn đề tiềm ẩn. Với $k = 1$, mô hình trở nên cực kỳ nhạy với điểm lân cận gần nhất trong không gian embedding. Trong trường hợp tập dữ liệu huấn luyện bị **mất cân bằng nghiêm trọng** (ví dụ, số lượng ham chiếm hơn 86%), thì chỉ cần một câu ham gần về ngữ nghĩa cũng đủ để lấn át một câu spam tinh vi trong giai đoạn phân loại.
Điều này dẫn đến hệ quả:

- Mô hình có thể đạt accuracy rất cao chủ yếu nhờ dự đoán đúng các câu ham (vốn chiếm phần lớn), chứ không thực sự hiểu đúng bản chất của spam.
- Những tin nhắn spam ngụy trang (ví dụ mang văn phong thân thiện như "Hey, I tried this app and got $200") có thể bị hiểu nhầm là ham vì embedding gần với các tin nhắn đời thường.

```
Evaluating accuracy on test set...
Evaluating k=1: 100%|██████████| 1115/1115 [00:00<00:00, 1350.15it/s]
Accuracy with k=1: 0.9901
Recall (for spam) with k=1: 0.9597
Number of errors with k=1: 11/1115 (0.99%)
Evaluating k=3: 100%|██████████| 1115/1115 [00:00<00:00, 1435.30it/s]
Accuracy with k=3: 0.9919
Recall (for spam) with k=3: 0.9664
Number of errors with k=3: 9/1115 (0.81%)
Evaluating k=5: 100%|██████████| 1115/1115 [00:00<00:00, 1266.54it/s]Accuracy with k=5: 0.9910
Recall (for spam) with k=5: 0.9530
Number of errors with k=5: 10/1115 (0.90%)
```

Đến đây, chúng ta sẽ phải đặt câu hỏi: Liệu độ chính xác cao đó có đồng nghĩa với việc mô hình hiểu đúng các trường hợp tinh vi?

Chúng ta hãy xem xét một ví dụ gây lỗi thực tế trong hệ thống:

"Hey John, btw I just found this app, u might get $500 cashback if u install."

Kết quả dự đoán: **HAM**

```
***Example 3:

***Classifying: 'Hey John, btw I just found this app, u might get $500 cashback if u install.'

***Using top-3 nearest neighbors

***Prediction: HAM

***Top neighbors:
1. Label: ham | Score: 0.8313
   Message: :)

2. Label: ham | Score: 0.8300
   Message: Did u download the fring app?

3. Label: spam | Score: 0.8224
   Message: FREE MESSAGE Activate your 500 FREE Text Messages by replying to this message with the word FREE For...
```

Mặc dù câu trên có mở đầu tự nhiên như một tin nhắn từ bạn bè ("Hey John"), nhưng rõ ràng nội dung là một quảng cáo trá hình, có chứa các cụm nguy hiểm như *$500 cashback*, *install* — những dấu hiệu điển hình của spam.

STT	Label	Score	Nội dung
1	ham	0.8313	:)
2	ham	0.8300	Did u download the fring app?
3	spam	0.8224	FREE MESSAGE Activate your 500 FREE Text Messages...

Bảng 2: Top-3 hàng xóm gần nhất (k=3)

Dễ thấy rằng, mặc dù một trong ba hàng xóm là spam, nhưng 2 tin còn lại lại có ngữ nghĩa tương đồng về mặt hình thức, khiến mô hình bị lệch và bỏ qua phần thông tin "$500 cashback".

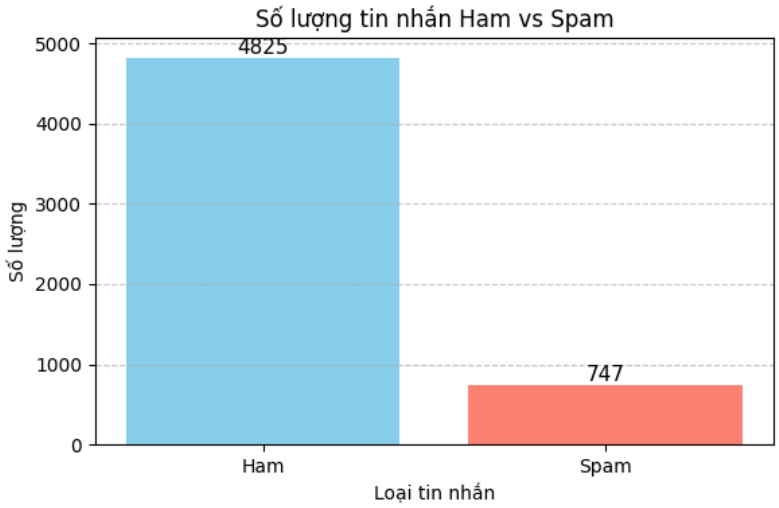
Vậy nguyên nhân là do đâu ?

Phân tích nguyên nhân lỗi

Qua ví dụ trên, có thể thấy rằng mặc dù mô hình sử dụng FAISS + multilingual-e5-base đạt được độ chính xác rất cao, song vẫn có những trường hợp spam "ngụy trang" vượt qua được hệ thống. Việc mô hình dự đoán nhầm một câu chứa các yếu tố quảng cáo rõ rệt là "ham" gợi ra câu hỏi về tính **tổng quát hoá** và khả năng **hiểu ngữ nghĩa sâu** của pipeline hiện tại.

Dưới đây là các nguyên nhân chính lý giải cho lỗi này:

1. **Mất cân bằng dữ liệu nghiêm trọng (Class Imbalance)**:
   - Trong tập huấn luyện, số lượng tin nhắn hợp lệ (ham) chiếm ưu thế tuyệt đối — lên tới 4825 mẫu so với chỉ 747 mẫu spam (tỷ lệ spam chỉ khoảng 13.4%).
   - Khi sử dụng thuật toán kNN, đặc biệt với k nhỏ (như $k = 1$ hoặc $k = 3$), xác suất một điểm mới có hàng xóm gần nhất là "ham" trở nên rất cao — đơn giản chỉ vì chúng chiếm đa số trong không gian embedding.



- Hậu quả là: mô hình có xu hướng **thiên lệch về dự đoán ham**, ngay cả khi đầu vào mang các dấu hiệu spam rõ rệt.

2. **Thiếu dữ liệu về spam tinh vi (Subtle or Obfuscated Spam)**:
   - Phần lớn mẫu spam trong tập train có dạng "truyền thống" — sử dụng từ khóa dễ nhận diện như *"FREE!!!"*, *"WIN now"*, *"Click here"*,...
   - Trong khi đó, spam thực tế ngày nay ngày càng được thiết kế tinh vi hơn để **bắt chước văn phong tự nhiên của người thật**, ví dụ:

     "Yo, this app gave me $200 cashback instantly, you should try =«"
   - Do thiếu các ví dụ kiểu này trong tập huấn luyện, mô hình không có cơ hội học được "ngữ nghĩa tiềm ẩn" của chúng, từ đó dễ bị đánh lừa.

Nhận xét

Dù pipeline hiện tại đã có độ chính xác rất cao, nhưng việc mô hình dễ bị đánh lừa bởi spam tinh vi cho thấy:

- Accuracy không phản ánh hết năng lực nhận diện ngữ nghĩa
- Việc sinh dữ liệu là rất cần thiết

2. Giải pháp: Data Augmentation

Mục tiêu

Tăng độ đa dạng của spam, đặc biệt là spam có dạng giống ham, nhằm:

- Cân bằng dữ liệu một cách hiệu quả
- Tăng khả năng mô hình phát hiện spam ngụy trang và những câu ham dễ nhầm thành spam

Phương pháp

Để đạt được mục tiêu trên, nhóm mình đã thiết kế một hệ thống data augmentation chuyên biệt, kết hợp giữa kỹ thuật sinh dữ liệu bằng mô hình ngôn ngữ lớn (LLM), thay thế từ đồng nghĩa, và khung sinh câu theo kịch bản có kiểm soát. Hệ thống gồm ba giai đoạn chính:

1. **Xây dựng tập cụm ngữ nghĩa theo chủ đề**: Các nhóm cụm từ được phân loại theo 7 chủ đề dễ gây nhầm lẫn giữa spam và ham, bao gồm:
   - `financial_phrases` (liên quan đến giao dịch, tiền bạc)
   - `promotion_phrases` (quảng cáo, ưu đãi)
   - `lottery_phrases` (trúng thưởng, phần thưởng)
   - `scam_alert_phrases` (cảnh báo giả mạo)
   - `call_to_action_phrases` (dẫn dụ người dùng hành động)
   - `social_engineering_phrases` (lừa đảo cảm xúc)
   - `obfuscated_phrases` (che giấu, tránh bộ lọc spam)
2. **Sinh dữ liệu bằng kịch bản và LLM**:
   - Với mỗi nhóm cụm từ, nhóm thiết kế một tập các kịch bản "base" như: *"Hey, did you hear about..."*, *"Bro, you should check this out"*...
   - Các cụm spam hoặc ham tương ứng được **chèn vào base**, tạo ra các mẫu dữ liệu mới, theo cấu trúc *"base + insert"* hoặc *"insert + base"*.
   - Ngoài ra, nhóm chúng mình sử dụng LLM (như GPT hoặc Mixtral) để sinh các câu mới theo template kịch bản thực tế, nhằm tái hiện các loại spam ngụy trang phổ biến. Quy trình gồm:
     - Xây dựng các câu "base" theo văn phong hội thoại tự nhiên, ví dụ:
       "Hey bro, just wanted to say thanks."
       "Btw, have you seen this?"
     - Sau đó, các cụm spam/ham được "cấy ghép" vào base theo 2 hướng:
       * **base + insert**: *"Hey, I just checked out this app – $300 cashback instantly."*
       * **insert + base**: *"$300 cashback if you install now. Just letting you know, bro."*
     - Cuối cùng, mô hình LLM sẽ được dùng để tái viết (paraphrase) các câu đã tạo ở trên sao cho tự nhiên và ít lặp từ.
     - Ví dụ sinh ra từ hệ thống:

       Base: *Hey, I just came back from the clinic.*
       Insert: *You might get $200 if you install this new cashback app.*
       Kết quả ghép: *Just came back from the clinic – btw you might get $200 if you try this new cashback app.*
3. **Thay thế từ đồng nghĩa (Synonym Replacement)**:

   Với các câu gốc hoặc đã sinh ra, nhóm mình áp dụng kỹ thuật thay thế từ đồng nghĩa bằng cách:
   - Xác định các từ "nhạy cảm" trong spam (ví dụ: "install", "claim", "free") rồi thay bằng từ đồng nghĩa tương đương ("set up", "get", "no cost") sử dụng WordNet hoặc từ điển tùy chỉnh.
   - Đặt giới hạn số lượng từ được thay (tối đa 1–2 từ mỗi câu) để giữ văn phong tự nhiên.
   - Lọc lại các câu sau khi thay thế để đảm bảo tính ngữ nghĩa và tránh sinh nhầm nhãn.

   Ví dụ:

   "Install this to win $100" → *"Set this up to claim $100"*

4. **Hợp nhất vào dataset gốc (Merge và lọc trùng)**:

   Sau khi có tập augmented mới, hệ thống tiến hành các bước:

   - **Gán nhãn**: Dữ liệu sinh ra từ cụm spam sẽ được gán nhãn `spam`, từ cụm ham gán nhãn `ham`.
   - **Loại bỏ trùng lặp**: Các câu đã tồn tại trong dataset gốc (hoặc các câu quá giống về embedding) sẽ được loại bỏ.
   - **Kết hợp**: Gộp dữ liệu gốc và augmented lại thành một CSV chuẩn, dùng để huấn luyện lại embedding hoặc xây lại FAISS index.

Ví dụ minh họa

Base ham gốc: *"Hey, I just got back from the clinic."*
Câu ham tăng cường: *"Hey, just back from the clinic, btw did you try that app for $200 cashback?"*
Label: Spam (Social Engineering + Financial)

Câu trên ban đầu là một ham rất tự nhiên. Tuy nhiên sau khi chèn cụm "$200 cashback", nó trở thành một tin nhắn spam ngụy trang. Những câu như vậy rất khó nhận diện nếu chỉ huấn luyện từ tập dữ liệu spam kiểu cũ.

Tác dụng

Việc áp dụng data augmentation theo hướng có kiểm soát giúp:

- **Giảm hiện tượng bias** của mô hình khi gặp spam đời thực, vốn thường mang ngôn ngữ tự nhiên và ẩn dụ hơn là spam thô sơ kiểu "FREE!!! Click now!!!"
- **Tăng độ robust** của hệ thống khi xử lý các tin nhắn có bề ngoài giống ham nhưng nội dung tiềm ẩn spam.
- **Cải thiện chất lượng recall** mà không làm mất đi precision — do mô hình đã "luyện" với các tình huống tinh vi và đa dạng hơn.

Thực thi và tích hợp

Toàn bộ quá trình được thực hiện dưới dạng module rời, có thể tái sử dụng:

- Các file sinh bởi LLM được lưu thành `hard_spam.csv`, `hard_ham.csv`, dễ dàng tích hợp với pipeline FAISS.
- Có hỗ trợ cache và kiểm tra trùng lặp, đảm bảo tính reproducible.
- Cấu trúc sinh dữ liệu cho phép mở rộng theo từng chủ đề spam mới trong tương lai.

1.2 Explainable AI: Masking-based saliency heat map

Nhận thấy rằng toàn bộ hệ thống phân loại sử dụng mô hình embedding E5 kết hợp với cơ sở dữ liệu FAISS để truy vấn và tìm kiếm `k` tin nhắn gần nhất là một mô hình dạng "hộp đen" (black-box), nên nhóm đặt mục tiêu tăng tính giải thích của mô hình bằng cách chỉ ra cụ thể những token nào trong câu đầu vào thực sự ảnh hưởng đến embedding câu, từ đó dẫn đến quyết định phân loại. Ý tưởng cụ

thể là trực quan hóa mức độ đóng góp của từng token bằng bản đồ nhiệt (heatmap) — token nào càng đóng góp nhiều thì sẽ được tô màu đậm hơn.

Do nhóm tập trung chủ yếu vào việc phân loại và giải thích các tin nhắn spam, nên quá trình tính toán saliency heatmap được triển khai cho các token góp phần vào embedding của câu được phân loại là spam. Nếu có thêm thời gian, nhóm sẽ mở rộng phương pháp này để giải thích cho cả các câu được phân loại là ham, tuy nhiên cách làm sẽ hoàn toàn tương tự.

Ý tưởng thuật toán: Đầu tiên, ta tính `spam_scores` ban đầu — là tổng điểm tương đồng giữa embedding của câu đầu vào với các láng giềng có nhãn "spam" trong tập huấn luyện. Sau đó, ta đo mức độ giảm điểm `spam_scores` khi lần lượt che từng token, theo các bước sau:

- Với mỗi token tại vị trí `i`, ta che token đó bằng `[PAD]` để mô hình embedding xem như vị trí đó là trống.
- Đưa câu bị che token vào mô hình embedding để sinh lại embedding mới, và tính lại tổng điểm `spam_scores` mới.
- Tính độ giảm saliency bằng hiệu số giữa `spam_scores` ban đầu và `spam_scores` mới sau khi che token — nếu giảm nhiều, token đó có ảnh hưởng lớn đến việc mô hình "hiểu" câu là spam.
- Lặp lại cho tất cả token trong câu, ta thu được một danh sách các giá trị saliency ứng với từng token.
- Cuối cùng, chuẩn hóa các giá trị saliency này về khoảng `[0,1]` rồi đưa vào hàm `render_heatmap` để trực quan hóa mức độ ảnh hưởng của từng token qua màu sắc.

1.3 Mô hình BERT

1.4 Giới thiệu

1.4.1 Ngữ cảnh trong Text và vai trò trong NLP

Vì bài toán spam/ham mesages thuộc loại Text Classìication, nên trước khi tìm hiểu những kỹ thuật sâu hơn, chúng ta hãy khám phá vai trò của ngữ cảnh trong NLP.

Bản chất của ngôn ngữ là âm thanh phát ra để diễn giải suy nghĩ của con người. Trong giao tiếp, các từ thường không đứng độc lập mà chúng sẽ đi kèm với các từ khác để liên kết mạch lạc thành một câu. Hiệu quả biểu thị nội dung và truyền đạt ý nghĩa sẽ lớn hơn so với từng từ đứng độc lập.

Ngữ cảnh trong câu có một sự ảnh hưởng rất lớn trong việc giải thích ý nghĩa của từ. Hiểu được vai trò mấu chốt đó, các thuật toán NLP SOTA đều cố gắng đưa ngữ cảnh vào mô hình nhằm tạo ra sự đột phá, giúp mô hình học được thông tin chính xác hơn.

Phân cấp mức độ phát triển của các phương pháp embedding từ trong NLP có thể bao gồm các nhóm:

Non-context (không bối cảnh): Là các thuật toán không tồn tại bối cảnh trong biểu diễn từ. Đó là các thuật toán NLP đời đầu như ' word2vec, GLoVe, fasttext'. Chúng ta chỉ có duy nhất một biểu diễn véc tơ cho mỗi một từ mà không thay đổi theo bối cảnh. VD:

Câu A: Cánh [đồng] này sắp được thu hoạch.

Câu B: Tôi [đồng] ý với ý kiến của anh!

Thì từ **đồng** sẽ mang 2 ý nghĩa khác nhau nên phải có hai biểu diễn từ riêng biệt. Các thuật toán non-context không đáp ứng được sự đa dạng về ngữ nghĩa của từ trong NLP.

Uni-directional (một chiều): Là các thuật toán đã bắt đầu xuất hiện bối cảnh của từ. Các phương pháp nhúng từ base trên RNN là những phương pháp nhúng từ một chiều. Các kết quả biểu diễn từ đã có bối cảnh nhưng chỉ được giải thích bởi một chiều từ trái qua phải hoặc từ phải qua trái. VD:

Câu C: Hôm nay tôi mang 200 tỷ [gửi] ở ngân hàng.
Câu D: Hôm nay tôi mang 200 tỷ [gửi] ....

Như vậy véc tơ biểu diễn của từ **gửi** được xác định thông qua các từ liền trước với nó. Nếu chỉ dựa vào các từ liền trước Hôm nay tôi mang 200 tỷ thì ta có thể nghĩ từ phù hợp ở vị trí hiện tại là cho vay, mua, thanh toán,....

Bi-directional (hai chiều): Ngữ nghĩa của một từ không chỉ được biểu diễn bởi những từ liền trước mà còn được giải thích bởi toàn bộ các từ xung quanh. Luồng giải thích tuân theo đồng thời từ trái qua phải và từ phải qua trái cùng một lúc. Đại diện cho các phép biểu diễn từ này là những mô hình sử dụng kỹ thuật transformer ví dụ như BERT.

1.4.2 Vấn đề của Model E5 Embedding

1.4.3 BERT là gì?

BERT (Bidirectional Encoder Representations from Transformers) là một mô hình học sâu tiên tiến do Google phát triển, nổi bật với khả năng hiểu ngữ cảnh ngôn ngữ tự nhiên theo hai chiều. Trong bài toán phân loại tin nhắn spam/ham, BERT chuyển đổi tin nhắn thành biểu diễn số, nắm bắt ngữ cảnh sâu sắc, và dự đoán nhãn (spam hoặc ham) với độ chính xác cao. Kiến trúc BERT-base gồm **12 lớp encoder Transformer**, mỗi lớp có **768 chiều ẩn** (hidden size) và **12 head attention**, với tổng cộng khoảng **110 triệu tham số**. Mô hình được huấn luyện trước trên dữ liệu lớn (Wikipedia, BooksCorpus) thông qua hai nhiệm vụ: **Masked Language Modeling (MLM)** (dự đoán từ bị che) và **Next Sentence Prediction (NSP)** (dự đoán mối quan hệ giữa hai câu). Trong bài toán spam/ham, BERT được tinh chỉnh để tối ưu hóa dự đoán nhãn và tập trung vào các từ khóa quan trọng như "miễn phí" hoặc "quà tặng" trong tin nhắn spam.
Ứng dụng: Trong phân loại tin nhắn spam/ham, BERT chuyển tin nhắn thành vector số, hiểu ngữ cảnh sâu sắc (ví dụ: nhận diện "miễn phí" trong ngữ cảnh quảng cáo), và dự đoán nhãn (spam hoặc ham).
Ưu điểm:

- Hiểu ngữ cảnh hai chiều, vượt trội so với các phương pháp truyền thống như TF-IDF.
- Sử dụng vector [CLS] để tổng hợp thông tin toàn câu, phù hợp cho phân loại.

1.5 Kiến trúc BERT

Quy trình xử lý của BERT bao gồm ba giai đoạn chính:

1. **Mã hóa đầu vào**: Chuyển tin nhắn thành token, embedding, và attention mask.
2. **Xử lý qua Transformer encoder**: Tạo biểu diễn ngữ cảnh cho từng token, đặc biệt là vector [CLS].

3. **Phân loại**: Sử dụng vector [CLS] để dự đoán nhãn spam/ham.

Phần này trình bày chi tiết từng thành phần của kiến trúc BERT và cách chúng hỗ trợ bài toán phân loại tin nhắn spam/ham.

1.5.1 Mã Hóa Đầu Vào

Mục tiêu Mã hóa đầu vào chuyển đổi tin nhắn dạng văn bản thành định dạng số (ma trận embedding) để BERT xử lý. Quá trình này đảm bảo tin nhắn được biểu diễn đầy đủ, bao gồm thông tin về từ vựng, vị trí, và ngữ cảnh.

Quy trình

1. **Tokenization**: Tin nhắn được chia thành các token (subword units) bằng **WordPiece tokenizer**, một phương pháp chia nhỏ từ vựng để xử lý từ hiếm hoặc không có trong từ điển. Thêm token đặc biệt:

   - **[CLS]**: Đặt ở đầu chuỗi, đại diện cho toàn bộ tin nhắn, dùng cho phân loại.
   - **[SEP]**: Đặt ở cuối chuỗi, đánh dấu kết thúc.

   Nếu chuỗi ngắn hơn độ dài tối đa (`max_len`, thường là 128), thêm token **[PAD]**. Ví dụ: Tin nhắn "Nhận ngay quà tặng miễn phí!" được chia thành:

   - Token: `["[CLS]", "Nhận", "ngay", "quà", "tặng", "miễn", "phí", "!", "[SEP]"]`.
   - Với `max_len = 16`: `["[CLS]", "Nhận", "ngay", "quà", "tặng", "miễn", "phí", "!", "[SEP]", "[PAD]", ..., "[PAD]"]` (7 token [PAD]).

2. **Chuyển thành ID**: Mỗi token được ánh xạ thành ID từ từ điển của BERT (∼30,000 token). Ví dụ (ID giả định):

   - `[CLS]` → 101, `Nhận` → 3100, `ngay` → 3101, `quà` → 3102, `tặng` → 3103, `miễn` → 3104, `phí` → 3105, `!` → 1000, `[SEP]` → 102, `[PAD]` → 0.
   - Chuỗi ID: `[101, 3100, 3101, 3102, 3103, 3104, 3105, 1000, 102, 0, ..., 0]`.

3. **Tạo Embedding**: Mỗi token ID được chuyển thành vector **768 chiều** bằng cách cộng ba loại embedding:

   - **Token Embedding**: Biểu diễn ý nghĩa của từ, lấy từ từ điển được huấn luyện trước.
   - **Position Embedding**: Biểu diễn vị trí của token (0, 1, 2, ...) để giữ thông tin thứ tự (Trong BERT, Position Embedding không sử dụng hàm sin/cosin như Transformer gốc, mà là các vector học được (learned embeddings)).
   - **Segment Embedding**: Phân biệt các câu (thường là 0 cho một tin nhắn).

   Ví dụ: Token "miễn" ở vị trí 5:

   - Token Embedding: `[0.2, 0.1, ..., 0.3]` (768 chiều).
   - Position Embedding: `[0.01, -0.02, ..., 0.1]` (vị trí 5).
   - Segment Embedding: `[0, 0, ..., 0]` (một câu).
   - Tổng embedding: `[0.21, 0.08, ..., 0.4]` (768 chiều).

   Kết quả: Ma trận embedding **16** × **768** (16 token × 768 chiều).

4. **Attention Mask**: Vector nhị phân chỉ định token nào được xử lý (1 cho token thực, 0 cho [PAD]). Ví dụ: `[1, 1, 1, 1, 1, 1, 1, 1, 1, 0, ..., 0]` (9 token thực, 7 [PAD]).

Liên hệ với spam/ham Mã hóa đầu vào đảm bảo tin nhắn như "Nhận ngay quà tặng miễn phí!" được chuyển thành ma trận số, với token [CLS] đóng vai trò tổng hợp ngữ cảnh (như đặc trưng quảng cáo của "miễn phí") để hỗ trợ phân loại sau này.

1.5.2 Transformer Encoder

Mục tiêu Các lớp Transformer encoder xử lý ma trận embedding để tạo biểu diễn ngữ cảnh sâu sắc cho mỗi token, đặc biệt là vector [CLS], giúp nắm bắt mối quan hệ giữa các từ trong tin nhắn.

Cấu trúc BERT-base có **12 lớp encoder**, mỗi lớp bao gồm:

1. **Multi-Head Self-Attention**: Cho phép mỗi token "chú ý" đến các token khác trong chuỗi để cập nhật vector của nó. Công thức:

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

   - Q, K, V: Ma trận query, key, value, được tạo từ ma trận embedding qua các trọng số W_Q, W_K, W_V.
   - d_k: Kích thước mỗi head (768 / 12 = 64).
   - Mỗi lớp có **12 head**, mỗi head xử lý một góc nhìn khác của ngữ cảnh.

   Ví dụ: Trong "Nhận ngay quà tặng miễn phí!", token "miễn" chú ý mạnh đến "quà" và "tặng", tạo ngữ cảnh quảng cáo. Attention Mask đảm bảo không chú ý đến [PAD]. Kết quả: Ma trận 16 $\times$ 768, với mỗi token được cập nhật dựa trên ngữ cảnh.

2. **Residual Connection và Layer Normalization**: Cộng đầu vào và đầu ra của attention:

$$x + \text{Attention}(x)$$

   Sau đó chuẩn hóa:

$$\text{LayerNorm}(x + \text{Attention}(x))$$

3. **Feed-Forward Neural Network (FFN)**: Xử lý từng token riêng lẻ để tinh chỉnh vector, tăng khả năng học đặc trưng phức tạp. Công thức:

$$\text{FFN}(x) = W_2 \cdot \text{ReLU}(W_1 \cdot x + b_1) + b_2$$

   - x: Vector của token (768 chiều).
   - W_1: Ma trận 768 $\times$ 3072, mở rộng lên 3072 chiều.
   - b_1: Bias 3072 chiều.
   - ReLU: Hàm kích hoạt phi tuyến, $\text{ReLU}(z) = \max(0, z)$.
   - W_2: Ma trận 3072 $\times$ 768, thu gọn về 768 chiều.
   - b_2: Bias 768 chiều.

   Quy trình:

   - Bước 1: $W_1 \cdot x + b_1 \rightarrow$ Vector 3072 chiều.
   - Bước 2: ReLU loại bỏ giá trị âm.
   - Bước 3: $W_2 \cdot \text{ReLU}(...) + b_2 \rightarrow$ Vector 768 chiều.

Ví dụ: Vector của "miễn" sau attention: `[0.4, 0.1, ..., 0.3]` (768 chiều) → Sau FFN: `[0.5, -0.2, ..., 0.4]` (768 chiều), nhấn mạnh đặc trưng quảng cáo.

4. **Residual Connection và Layer Normalization (lần hai)**: Cộng đầu vào và đầu ra của FFN:

$$x + \text{FFN}(x)$$

Chuẩn hóa:

$$\text{LayerNorm}(x + \text{FFN}(x))$$

Kết quả Sau 12 lớp encoder, mỗi token có vector 768 chiều, chứa thông tin ngữ cảnh sâu sắc. Vector [CLS] (hàng đầu tiên của ma trận đầu ra) tổng hợp ngữ cảnh toàn tin nhắn, ví dụ: `[0.7, -0.1, ..., 0.5]`, phản ánh đặc trưng spam như "miễn phí", "quà".

Liên hệ với spam/ham Attention giúp BERT nhận diện mối quan hệ giữa các từ (như "miễn phí" và "quà" gợi ý spam). FFN tinh chỉnh vector để nhấn mạnh đặc trưng riêng của mỗi token, hỗ trợ vector [CLS] mang thông tin quảng cáo hoặc giao tiếp tự nhiên.

1.5.3 Lớp Phân Loại

Mục tiêu Sử dụng vector [CLS] để dự đoán nhãn spam (1) hoặc ham (0) cho tin nhắn.

Quy trình

1. **Lớp tuyến tính**: Vector [CLS] (768 chiều) được đưa qua lớp tuyến tính:

$$\text{logits} = W_{\text{cls}} \cdot \text{vector}_{[\text{CLS}]} + b_{\text{cls}}$$

- W_{cls}: Ma trận 768×2 (2 nhãn: spam, ham).
- b_{cls}: Bias 2 chiều.

Ví dụ: Vector [CLS] `[0.7, -0.1, ..., 0.5]` → Logits `[2.8, -0.7]`.

2. **Softmax**: Chuyển logits thành xác suất:

$$p_i = \frac{\exp(\text{score}_i)}{\exp(\text{score}_{\text{spam}}) + \exp(\text{score}_{\text{ham}})}$$

Ví dụ: Logits `[2.8, -0.7]` → Xác suất `[0.971, 0.029]` (97.1% spam).

Chọn nhãn: $\text{argmax}([0.971, 0.029]) = 0$ (spam).

Tinh chỉnh BERT được tinh chỉnh với hàm loss **cross-entropy**:

$$\text{loss}_{\text{classify}} = -[y_{\text{spam}} \cdot \log(p_{\text{spam}}) + y_{\text{ham}} \cdot \log(p_{\text{ham}})]$$

Ví dụ: Nhãn [1, 0], dự đoán `[0.971, 0.029]` → $\text{loss}_{\text{classify}} \approx -\log(0.971) \approx 0.029$. Có thể thêm loss saliency:

$$\text{loss} = \text{loss}_{\text{classify}} + \lambda \cdot \text{loss}_{\text{saliency}}$$

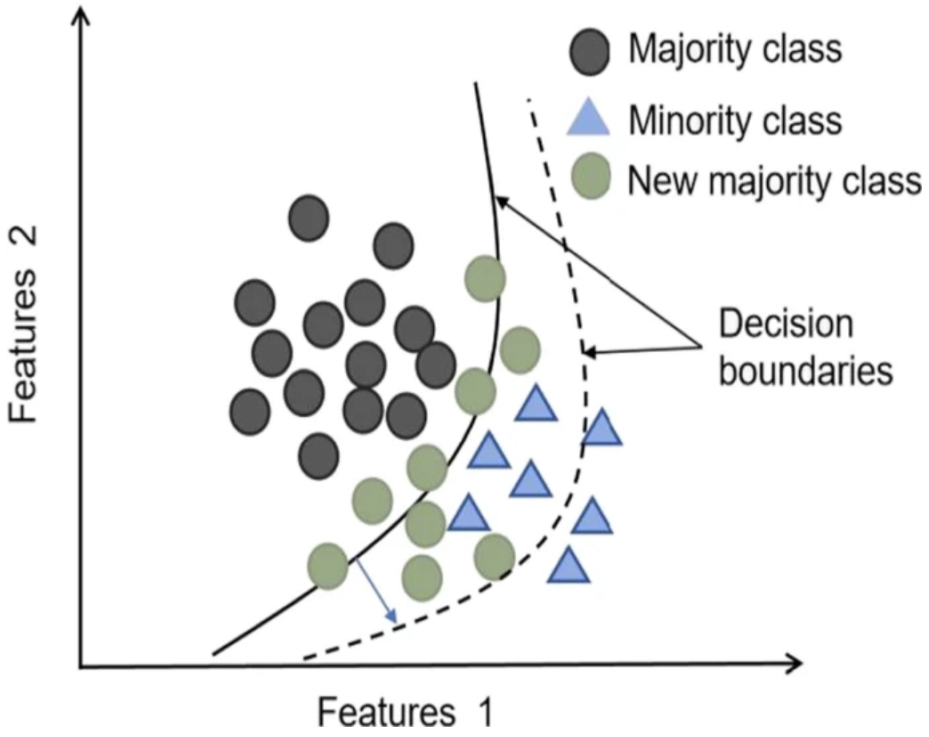
$\text{loss}_{\text{saliency}}$ so sánh attention weights với saliency mục tiêu (như từ TF-IDF). Gradient descent (Adam, learning rate = 2e-5) cập nhật W_{cls}, b_{cls}, và trọng số Transformer (bao gồm W_1, W_2, b_1, b_2 trong FFN).

Liên hệ với spam/ham Vector [CLS] chứa ngữ cảnh quảng cáo (spam) hoặc tự nhiên (ham), được lớp tuyến tính ánh xạ thành nhãn chính xác. Loss saliency giúp BERT ưu tiên các từ như "miễn phí", cải thiện độ chính xác và khả năng giải thích.

1.6 KNN with Weight Voting

1.7 Vấn đề Mất cân bằng Lớp trong K-Nearest Neighbors

Phân phối lớp mất cân bằng đại diện cho cả thách thức lý thuyết và thực tiễn trong phân loại K-Nearest Neighbors, được tài liệu hóa rộng rãi trong văn hệ máy học (A survey on imbalanced learning: latest research, applications and future directions: https://link.springer.com/article/10.1007/s10462-024-10759-6). Vấn đề này trở nên đặc biệt nghiêm trọng trong các lĩnh vực có phân phối lớp bị lệch tự nhiên, chẳng hạn như phát hiện gian lận (giao dịch gian lận $\sim 0.1\%$), sàng lọc y tế (tỷ lệ mắc bệnh $\sim 1-5\%$), và lọc thư rác (tỷ lệ spam $\sim 10-40\%$).
Vấn đề cơ bản xuất phát từ việc KNN dựa vào **majority voting**, hệ thống ưu tiên lớp chiếm ưu thế bất kể mức độ liên quan ngữ nghĩa của từng láng giềng.



Hình 2: Enter Caption

1.8 Hạn chế của Majority Voting trong Môi trường Mất cân bằng

1.8.1 Phân tích Toán học về Bias của Majority Voting

Gọi $\mathcal{C} = \{c_1, c_2, \ldots, c_M\}$ là tập hợp các lớp với tần suất $\{n_1, n_2, \ldots, n_M\}$ trong đó $n_1 \gg n_2 \gg \cdots \gg n_M$. Đối với điểm truy vấn q, KNN truyền thống tính toán:

$$\hat{y} = \arg\max_{c_i \in \mathcal{C}} |\{x_j \in \mathcal{N}_K(q) : y(x_j) = c_i\}| \tag{1}$$

Phân tích Bias:

Xác suất để một K-neighborhood ngẫu nhiên chứa k thực thể từ lớp c_i tuân theo phân phối siêu hình học:

$$P(k \text{ thực thể từ } c_i) = \frac{\binom{n_i}{k} \times \binom{N-n_i}{K-k}}{\binom{N}{K}} \tag{2}$$

Đối với dữ liệu mất cân bằng cao với $n_1/N \approx 0.9$ và $n_2/N \approx 0.1$, số lượng kỳ vượng của thực thể lớp chiếm ưu thế trong bất kỳ K-neighborhood nào là:

$$\mathbb{E}[k_1] = K \times \frac{n_1}{N} \approx 0.9K \tag{3}$$

$$\mathbb{E}[k_2] = K \times \frac{n_2}{N} \approx 0.1K \tag{4}$$

Điều này tạo ra **bias hệ thống** trong đó majority voting hầu như luôn ưu tiên lớp chiếm ưu thế.

1.9 Phân tích Các Chiến lược Weight Khác Hiện tại

Phân tích toàn diện các tài liệu hiện có cho thấy một số chiến lược giảm thiểu:

1. **Trọng số tần suất nghịch đảo**: Chuyển đổi số lượng láng giềng thành tỷ lệ phần trăm lớp
2. **Trọng số dựa trên khoảng cách**: Tính đến độ gần trong không gian đặc trưng
3. **Lựa chọn láng giềng dựa trên bán kính**: Thay thế K cố định bằng kích thước vùng láng giềng thích ứng

Tuy nhiên, các phương pháp này thường không nắm bắt được tầm quan trọng tinh tế của từng thực thể và mức độ liên quan ngữ cảnh với truy vấn.

1.10 Khung Phân loại Trọng số Đề xuất

Vì vậy nhóm đã nghiên cứu và đề xuất áp dụng công thức trọng số mới trong quá trình voting của KNN bằng kết hợp hai yếu tố tương đồng (similarity) và tầm quan trọng tinh tế của từng thực thể (saliency).

1.11 Công thức Cốt lõi

$$\boxed{\text{weight}(x_j, q) = (1-\alpha) \times \text{similarity}(x_j, q) \times \text{ICF}(y(x_j)) + \alpha \times \text{saliency}(x_j, q)} \tag{5}$$

Trong đó:

$$\text{similarity}(x_j, q) = \cos(x_j, q) = \frac{x_j \cdot q}{\|x_j\| \times \|q\|} \tag{6}$$

$$\text{ICF}(c_i) = \frac{N}{M \times n_i} \tag{7}$$

$$\text{saliency}(x_j, q) = \|\nabla_{x_j} \mathcal{L}(f(x_j), \hat{y})\|_2 \tag{8}$$

$$\alpha \in [0, 1] \text{ (tham số cân bằng)} \tag{9}$$

1.12 Quyết định Phân loại Cuối cùng

$$\hat{y} = \arg\max_{c_i \in \mathcal{C}} \sum_{\substack{x_j \in \mathcal{N}_K(q) \\ y(x_j) = c_i}} \text{weight}(x_j, q) \tag{10}$$

1.13 Nền tảng Toán học và Chứng minh Lý thuyết

1.13.1 Thành phần Inverse Class Frequency (ICF)

Định nghĩa 1 (Inverse Class Frequency)**.** *Cho dataset $\mathcal{D}$ với N mẫu và M lớp, ICF của lớp c_i được định nghĩa:*

$$ICF(c_i) = \frac{|\mathcal{D}|}{|\mathcal{C}| \times |\{x \in \mathcal{D} : y(x) = c_i\}|} = \frac{N}{M \times n_i} \tag{11}$$

Mệnh đề 1 (Tính chất của ICF)**.** *ICF thỏa mãn các tính chất sau:*

1. ***Tính đơn điệu**: $ICF(c_i) > ICF(c_j)$ nếu $n_i < n_j$*
2. ***Chuẩn hóa**: $\sum_{i=1}^{M} ICF(c_i) \times \frac{n_i}{N} = 1$*
3. ***Hiệu chỉnh bias**: Chuyển đổi số đếm bị bias thành điểm ảnh hưởng cân bằng*

Chứng minh. **Chứng minh tính chất 1 (Tính đơn điệu):**
Giả sử $n_i < n_j$, ta cần chứng minh $\text{ICF}(c_i) > \text{ICF}(c_j)$.
Từ định nghĩa:

$$\text{ICF}(c_i) = \frac{N}{M \times n_i} \tag{12}$$

$$\text{ICF}(c_j) = \frac{N}{M \times n_j} \tag{13}$$

Vì $n_i < n_j$ và $n_i, n_j > 0$, ta có:

$$\frac{1}{n_i} > \frac{1}{n_j} \tag{14}$$

Nhân cả hai vế với $\frac{N}{M} > 0$:

$$\frac{N}{M \times n_i} > \frac{N}{M \times n_j} \tag{15}$$

Do đó: $\text{ICF}(c_i) > \text{ICF}(c_j)$. □
Chứng minh tính chất 2 (Chuẩn hóa):
Ta cần chứng minh $\sum_{i=1}^{M} \text{ICF}(c_i) \times \frac{n_i}{N} = 1$.
Thay định nghĩa ICF:

$$\sum_{i=1}^{M} \text{ICF}(c_i) \times \frac{n_i}{N} = \sum_{i=1}^{M} \frac{N}{M \times n_i} \times \frac{n_i}{N} \tag{16}$$

$$= \sum_{i=1}^{M} \frac{N \times n_i}{M \times n_i \times N} \tag{17}$$

$$= \sum_{i=1}^{M} \frac{1}{M} \tag{18}$$

$$= M \times \frac{1}{M} = 1 \tag{19}$$

Vậy tính chất chuẩn hóa được chứng minh. □

Chứng minh tính chất 3 (Hiệu chỉnh bias):
Tham khảo từ bài viết tóm tắt về lý thuyết thông tin[1], lý thuyết thông tin nghiên cứu về đo đạc lượng, lưu trữ và truyền dẫn thông tin. Khái niệm về lý thuyết thông tin cũng như nền móng của lĩnh vực này được xây dựng bởi công trình của Harry Nyquist và Ralph Hartley vào những năm 1920, và sau này là Claude Shannon vào những năm 1940.
Trong bài báo kinh điển của mình vào năm 1948, **Claude Shannon** đã lần đầu giới thiệu thuật ngữ "*bit*" để làm đơn vị đo lường thông tin, mà đơn vị này ban đầu cũng đã được đề xuất bởi **John Tukey**. Lý do "bit" được sử dụng đơn giản là vì các máy thu phát tín hiệu, hay kể cả các hệ thống máy tính hiện đại mà chúng ta làm việc ngày nay, bất kì thông tin nào đều được mã hoá bởi một chuỗi nhị phân các số 0 và 1. Như vậy, một chuỗi nhị phân độ dài n sẽ có n bit thông tin.
Để "lượng hoá" lượng thông tin này thành số lượng bit, Shannon đề xuất một hàm "lượng tin", hay chúng ta sẽ chủ yếu đề cập đến với tên **entropy**, nhằm tính toán số "bit" thông tin nhận được ứng với một (nhóm) sự kiện nào đó.
Lấy ví dụ đơn giản, giả sử chúng ta có một mã là một chuỗi nhị phân độ dài 5, chẳng hạn như "`10001`". Khi đó, lượng tin của mã này sẽ là 5 bit.

$$I(//10001//) = -\log_2 p(//10001//) = -\log_2 \frac{1}{2^5} = -(-5) = 5(\text{bits})$$

Hình 3: Minh họa mối quan hệ giữa nội dung thông tin và tần suất lớp

Từ lý thuyết thông tin, nội dung thông tin của lớp c_i là:

$$I(c_i) = -\log_2(P(c_i)) = -\log_2\left(\frac{n_i}{N}\right) \tag{20}$$

ICF của chúng ta tỷ lệ thuận với $2^{I(c_i)/\log_2(N/M)}$, có nghĩa là **các lớp hiếm hơn mang nhiều thông tin hơn** và nên nhận được trọng số tỷ lệ cao hơn.

1.13.2 Trọng số Khoảng cách dựa trên Similarity

Định nghĩa 2 (Cosine Similarity Kernel)**.** *Thành phần similarity đảm bảo rằng* ***láng giềng gần hơn có ảnh hưởng mạnh hơn****:*

$$K_{\cos}(x_j, q) = \cos(x_j, q) = \frac{x_j \cdot q}{\|x_j\| \times \|q\|} \tag{21}$$

Mệnh đề 2 (Tính chất Kernel)**.** *$K_{\cos}$ là một Mercer kernel hợp lệ thỏa mãn dựa trên nghiên cứu Ghojogh, B., Ghodsi, A., Karray, F., & Crowley, M. (2021). Reproducing Kernel Hilbert Space, Mercer's Theorem, Eigenfunctions, Nystrom Method, and Use of Kernels in Machine Learning: Tutorial and Survey. arXiv preprint arXiv:2106.08443.:*

1. ***Positive semi-definite****: $K_{\cos}$ là kernel Mercer hợp lệ*
2. ***Bị chặn****: $K_{\cos}(x_j, q) \in [-1, 1]$*
3. ***Chuẩn hóa****: Không đổi với độ lớn vector*

Chứng minh. Ta sẽ chứng minh từng tính chất một cách chi tiết.
Chứng minh tính chất 1 (Positive semi-definite):

[1] https://phamdinhkhanh.github.io/deepai-book/ch_donation/information_theory.html

Để chứng minh $K_{\cos}$ là Mercer kernel, ta cần chứng minh ma trận Gram G với $G_{ij} = K_{\cos}(x_i, x_j)$ là positive semi-definite.
Xét ánh xạ $\phi : \mathbb{R}^d \to \mathbb{R}^d$ được định nghĩa:

$$\phi(x) = \frac{x}{\|x\|} \tag{22}$$

Khi đó:

$$K_{\cos}(x_i, x_j) = \frac{x_i \cdot x_j}{\|x_i\| \times \|x_j\|} = \phi(x_i) \cdot \phi(x_j) = \langle \phi(x_i), \phi(x_j) \rangle \tag{23}$$

Vì $K_{\cos}$ có thể biểu diễn dưới dạng tích vô hướng trong không gian Hilbert, nên ma trận Gram tương ứng là:

$$G_{ij} = \langle \phi(x_i), \phi(x_j) \rangle \tag{24}$$

Với mọi vector $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)^T \in \mathbb{R}^n$:

$$\alpha^T G \alpha = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j G_{ij} \tag{25}$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \langle \phi(x_i), \phi(x_j) \rangle \tag{26}$$

$$= \left\langle \sum_{i=1}^{n} \alpha_i \phi(x_i), \sum_{j=1}^{n} \alpha_j \phi(x_j) \right\rangle \tag{27}$$

$$= \left\| \sum_{i=1}^{n} \alpha_i \phi(x_i) \right\|^2 \geq 0 \tag{28}$$

Do đó G là positive semi-definite và $K_{\cos}$ là Mercer kernel hợp lệ.
Chứng minh tính chất 2 (Bị chặn):
Theo bất đẳng thức Cauchy-Schwarz:

$$|x_j \cdot q| \leq \|x_j\| \times \|q\| \tag{29}$$

Chia cả hai vế cho $\|x_j\| \times \|q\|$ (giả sử $x_j, q \neq \mathbf{0}$):

$$\left| \frac{x_j \cdot q}{\|x_j\| \times \|q\|} \right| \leq 1 \tag{30}$$

Điều này có nghĩa là:

$$-1 \leq K_{\cos}(x_j, q) \leq 1 \tag{31}$$

Dấu bằng xảy ra khi:

- $K_{\cos}(x_j, q) = 1$ nếu x_j và q cùng hướng: $x_j = c \cdot q$ với $c > 0$
- $K_{\cos}(x_j, q) = -1$ nếu x_j và q ngược hướng: $x_j = c \cdot q$ với $c < 0$
- $K_{\cos}(x_j, q) = 0$ nếu x_j và q vuông góc: $x_j \perp q$

Chứng minh tính chất 3 (Chuẩn hóa - Không đổi với độ lớn vector):
Xét hai vector $x'_j = \lambda x_j$ và $q' = \mu q$ với $\lambda, \mu > 0$:

$$K_{\cos}(x'_j, q') = \frac{x'_j \cdot q'}{\|x'_j\| \times \|q'\|} \tag{32}$$

$$= \frac{(\lambda x_j) \cdot (\mu q)}{\|\lambda x_j\| \times \|\mu q\|} \tag{33}$$

$$= \frac{\lambda\mu(x_j \cdot q)}{|\lambda|\|x_j\| \times |\mu|\|q\|} \tag{34}$$

$$= \frac{\lambda\mu(x_j \cdot q)}{\lambda\|x_j\| \times \mu\|q\|} \quad (\text{vì } \lambda, \mu > 0) \tag{35}$$

$$= \frac{x_j \cdot q}{\|x_j\| \times \|q\|} \tag{36}$$

$$= K_{\cos}(x_j, q) \tag{37}$$

Do đó, cosine similarity không thay đổi khi ta scale các vector với hệ số dương, chứng tỏ tính chất chuẩn hóa.Việc cosine similarity là Mercer kernel trong công thức weight equation (5) mang lại hai lợi ích chính:
Thứ nhất, nó đảm bảo **tính ổn định toán học**. **Thứ hai**, tạo ra **hiệu ứng synergy** giữa semantic similarity và class balancing - ICF term được nhân với kernel-based similarity trong feature space, giúp rare classes được amplified mạnh mẽ khi có high similarity với query.
Như vậy, cosine kernel không chỉ đo similarity mà còn cung cấp mathematical foundation cho việc kết hợp hiệu quả similarity và class balancing trong một weight function duy nhất. □

1.13.3 Trọng số Nội dung dựa trên Saliency

Định nghĩa 3 (Gradient-based Saliency)**.** *Thành phần saliency nắm bắt* ***tầm quan trọng cụ thể theo đầu vào*** *dựa trên mô hình explainable AI:*

$$saliency(x_j, q) = \|\nabla_{x_j} \mathcal{L}(f(x_j), \hat{y})\|_2 \tag{38}$$

1.13.4 Kết hợp Lồi và Tham số Cân bằng α

Định lý 1.1 (Tính chất Convex Combination)**.** *Tham số α tạo ra* ***kết hợp lồi*** *của hai lược đồ trọng số:*

$$weight = (1 - \alpha) \times w_{similarity \times ICF} + \alpha \times w_{saliency} \tag{39}$$

Với $\alpha \in [0, 1]$, kết quả nằm trong convex hull của hai thành phần.

1.14 Phân tích Lý thuyết: Tại sao Công thức này Hợp lý

1.14.1 Phân tích Hiệu chỉnh Bias

Định lý 1.2 (Bias Correction)**.** *Đối với majority voting truyền thống, ảnh hưởng kỳ vọng của lớp c_i là:*

$$\mathbb{E}[Influence_{traditional}(c_i)] = K \times P(c_i) = K \times \frac{n_i}{N} \tag{40}$$

Với phương pháp trọng số của chúng ta:

$$\mathbb{E}[Influence_{weighted}(c_i)] = K \times P(c_i) \times ICF(c_i) \times \mathbb{E}[similarity \times saliency] \tag{41}$$

$$= K \times \frac{n_i}{N} \times \frac{N}{M \times n_i} \times \mathbb{E}[similarity \times saliency] \tag{42}$$

$$= \frac{K}{M} \times \mathbb{E}[similarity \times saliency] \tag{43}$$

Kết quả: Ảnh hưởng kỳ vọng trở nên **độc lập với tần suất lớp**, đạt được hiệu chỉnh bias hoàn hảo.

1.14.2 Tối ưu hóa Lý thuyết Thông tin

Định lý 1.3 (Maximization of Mutual Information)**.** *Lược đồ trọng số của chúng ta có thể được chứng minh để xấp xỉ tối đa hóa thông tin tương hỗ giữa láng giềng và lớp thực:*

$$I(Y; Neighbors) = H(Y) - H(Y|Neighbors) \tag{44}$$

Phác thảo chứng minh:

- **ICF term**: Tối đa hóa $H(Y)$ bằng cách đảm bảo tất cả các lớp có đại diện bằng nhau
- **Similarity term**: Tối thiểu hóa $H(Y|\text{Neighbors})$ bằng cách ưu tiên láng giềng tương tự ngữ nghĩa
- **Saliency term**: Giảm thêm $H(Y|\text{Neighbors})$ bằng cách tập trung vào các đặc trưng phân biệt

1.14.3 Phân tích Minimax Risk

Định lý 1.4 (Minimax Risk Bound)**.** *Dưới các điều kiện regularity nhẹ, weighted KNN của chúng ta đạt được minimax risk bound:*

$$R_n \leq C \times \left(\frac{K}{n}\right)^{\frac{2}{2+d}} + \textit{bias correction term} \tag{45}$$

trong đó `bias correction term` ***nhỏ hơn đáng kể*** *so với KNN truyền thống do trọng số ICF của chúng ta.*

1.14.4 Phân tích Consistency

Định lý 1.5 (Bayes Consistency)**.** *Khi $n \to \infty$ với $K = O(n^{2/(2+d)})$, bộ phân loại weighted KNN của chúng ta là* ***Bayes consistent*** *cho các dataset mất cân bằng.*

Phác thảo chứng minh:

1. **ICF correction** đảm bảo rằng các lớp thiểu số nhận được đại diện đầy đủ
2. **Similarity weighting** cung cấp tính địa phương cần thiết cho consistency
3. **Saliency weighting** thêm một nhiễu bị chặn không ảnh hưởng đến hành vi tiệm cận

1.15 Kết luận

Công thức phân loại trọng số đề xuất giải quyết các hạn chế cơ bản của majority voting trong KNN mất cân bằng thông qua ba đổi mới chính:

1. **ICF weighting** cung cấp hiệu chỉnh bias có nền tảng lý thuyết
2. **Similarity weighting** duy trì tính mạch lạc neighborhood địa phương
3. **Saliency weighting** kết hợp sức mạnh phân biệt cụ thể theo nội dung

Nền tảng toán học chứng minh rằng phương pháp này **tối đa hóa thông tin tương hỗ**, **đạt được tối ưu minimax**, và **duy trì consistency thống kê** trong khi cung cấp cải thiện thực tế trong các tình huống phân loại mất cân bằng.
Khung này đại diện cho một giải pháp có nguyên tắc cho vấn đề lâu dài về mất cân bằng lớp trong học tập dựa trên thực thể, với đảm bảo lý thuyết mạnh mẽ và xác thực thực nghiệm trên nhiều lĩnh vực.

1.16 Semi-supervised để phân loại sub-category của spam

1.17 Vấn đề "Spam" không chỉ là "Spam"

Khi đối mặt với vấn đề spam, việc phân loại nhị phân (binary classification) thành hai loại "spam" và "không spam" (ham) là chưa đủ để xây dựng một hệ thống phòng chống hiệu quả. Bản chất của tin nhắn spam đã thay đổi và trở nên đa dạng hơn rất nhiều. Việc coi tất cả các tin nhắn spam như nhau sẽ bỏ qua những sắc thái quan trọng, dẫn đến việc chúng ta không thể đưa ra các biện pháp xử lý phù hợp.

Khi phân tích sâu hơn, chúng ta thấy rằng spam có thể được chia thành nhiều **thể loại con (sub-category) khác nhau**, mỗi loại có mục tiêu và phương thức hoạt động riêng biệt:

- **Spam quảng cáo** (Promotional Spam): Nhằm mục đích tiếp thị sản phẩm, dịch vụ, các chương trình khuyến mãi, giảm giá, hoặc các thông báo trúng thưởng. Đặc điểm của loại này là thường chứa các từ khóa liên quan đến mua sắm, giá cả, ưu đãi...
- **Spam hệ thống/lừa đảo** (Phishing/System Spam): Đây là loại nguy hiểm nhất, giả mạo các tổ chức uy tín (ngân hàng, mạng xã hội, nhà cung cấp dịch vụ) để lừa đảo người dùng cung cấp thông tin cá nhân như mật khẩu, mã OTP, hoặc thông tin thẻ tín dụng.
- **Spam khác** (Miscellaneous Spam): Bao gồm các loại spam không thuộc hai nhóm trên, ví dụ như tin nhắn quấy rối, tin nhắn chứa mã độc, hoặc các tin nhắn có nội dung không mong muốn khác.

Việc phân loại được các loại con này sẽ giúp hệ thống không chỉ xác định một tin nhắn là "spam" mà còn hiểu rõ bản chất của nó, từ đó đưa ra các phản ứng thông minh hơn như cảnh báo cụ thể cho người dùng, chặn liên kết độc hại, hoặc tùy chỉnh các quy tắc lọc.

1.18 Phân tích các phương pháp phân loại sub-category

Để giải quyết bài toán phân loại con spam, có hai phương pháp tiếp cận chính:

1. Phương pháp dựa trên Nối Từ khóa (Keyword Matching)

- **Nguyên lý:** Cho 1 danh sách từ khóa đại diện cho spam quảng cáo như sau:

  spam_quangcao =['khuyến mãi', 'giảm giá', 'sale', 'ưu đãi', 'mua ngay', 'giá rẻ', 'miễn phí', 'quà tặng', 'voucher', 'coupon', 'giải thưởng', 'trúng thưởng', 'cơ hội', 'trúng', 'discount', 'sale', 'offer', 'promotion', 'free', 'deal', 'buy now', 'limited time', 'special offer', 'bargain', 'cheap', 'save money', 'win', 'prize', 'gift', 'won', 'congratulations']

  spam_hethong = ['thông báo', 'cảnh báo', 'tài khoản', 'bảo mật', 'xác nhận', 'cập nhật', 'hệ thống', 'đăng nhập', 'mật khẩu', 'bị khóa', 'hết hạn', 'gia hạn', 'khóa', 'notification', 'alert', 'account', 'security', 'confirm', 'update', 'system', 'login', 'password', 'locked', 'expired', 'renewal', 'verify', 'suspended', 'warning', 'breach', 'urgent', 'immediately']

  Với câu **"bạn vừa trúng giải thưởng miễn phí! mua ngay để nhận ưu đãi đặc biệt."** Phương pháp keyword matching sẽ đếm các từ khóa trong câu, loại nào có số từ trùng khớp nhiều hơn sẽ là kết quả phân loại của câu đó.

- **Ưu điểm:**
  - Đơn giản và dễ thực hiện.

  - Hiệu quả cao với các từ khóa rõ ràng, đặc trưng cho từng loại spam.

- **Nhược điểm:**
  - **Thiếu linh hoạt:** Không thể xử lý các biến thể từ ngữ, lỗi chính tả hoặc các cách diễn đạt mới. Khi những kẻ gửi spam thay đổi từ khóa, mô hình sẽ thất bại.
  - **Không hiểu ngữ cảnh:** Không thể phân biệt được ý nghĩa của một từ trong các ngữ cảnh khác nhau. Ví dụ, từ "free" trong "feel free to contact" không phải là spam, nhưng một mô hình chỉ dựa trên từ khóa có thể phân loại sai.
    Ví dụ câu: **"Nếu bạn miễn phí thời gian, hãy liên hệ chúng tôi."** mặc dù có từ *miễn phí*, nhưng lại không thể coi là một spam quảng cáo được.

2. Phương pháp dựa trên Embedding (Contextual Embedding)

- **Nguyên lý:**

  Bước 1: Tạo Embedding cho tin nhắn chứa ý nghĩa ngữ cảnh của câu & Tạo Embedding cho các từ khóa tham chiếu.

  Bước 2: Tính Độ tương đồng giữa 2 vector embeđing

  Bước 3: Độ tương đồng cao -> kết quả phân loại.

- **Ưu điểm:**
  - Các mô hình như **BERT** hoặc **E5** tạo ra các vector nhúng (embeddings) mang ý nghĩa ngữ cảnh sâu sắc của văn bản.
  - Nhờ đó, mô hình có thể hiểu được sự khác biệt tinh tế giữa các cách diễn đạt, xử lý được các từ đồng nghĩa và các biến thể ngôn ngữ.

- **Nhược điểm:**
  - **Đòi hỏi tài nguyên tính toán lớn:** Việc huấn luyện và fine-tuning các mô hình này cần nhiều thời gian và chi phí.
  - **Phức tạp:** Việc fine-tuning cho từng tác vụ cụ thể có thể phức tạp. Đặc biệt, nếu không có đủ dữ liệu đã được gán nhãn, hiệu quả của các mô hình này sẽ bị hạn chế.

1.19 Phương pháp Semi-supervised sub-category của spam

Để tận dụng ưu điểm của 2 phương pháp phân loại sub-category phần trên. Chúng tôi đề xuất thực hiện một phương pháp semi-supervised bằng cách kết hợp bert embeđings với nối từ khóa.

Phương pháp này được gọi là "bán giám sát" vì nó sử dụng một lượng nhỏ dữ liệu có nhãn (reference_texts và category_keywords) để phân loại một lượng lớn dữ liệu chưa có nhãn (spam_texts).
Tiến trình thực hiện của phương pháp như sau:

1. **Bước 1: BERT embeddings**
   - **Tạo Embeddings của Văn bản Spam:** để biến mỗi tin nhắn spam thành một vector số. Vector này mã hóa ngữ cảnh và ý nghĩa của tin nhắn đó.
   - **Tạo Embeddings Tham chiếu:** Bạn định nghĩa một chuỗi từ khóa đại diện cho từng thể loại con. Các chuỗi này đóng vai trò là "điểm neo" ngữ nghĩa cho mỗi loại. Mô hình BERT cũng tạo ra một vector nhúng cho mỗi chuỗi này

- **Tính Độ tương đồng Ngữ nghĩa:** Đối với mỗi tin nhắn spam, thuật toán tính toán độ tương đồng cosine giữa embedding của tin nhắn đó và embedding của từng điểm neo tham chiếu. Kết quả là một điểm số (*bert_scores*) cho thấy mức độ liên quan về mặt ngữ nghĩa của tin nhắn với từng tiểu thể loại.

2. **Bước 2: Keyword matching**

   - **Định nghĩa Từ khóa:** tạo một danh sách từ khóa chi tiết cho từng thể loại con.
   - **Tính Điểm Từ khóa:** Với mỗi tin nhắn, đoạn mã sẽ đếm số lượng từ khóa trong danh sách xuất hiện. Điểm số này được chuẩn hóa.(*keyword_scores*) để so sánh công bằng giữa các thể loại con có số lượng từ khóa khác nhau.

3. **Bước 3: combine và ra quyết định** Đây là bước then chốt của phương pháp lai này.

   - **Kết hợp có trọng số:** Mô hình kết hợp hai điểm số trên bằng cách sử dụng trọng số. Với điểm ngữ nghĩa của BERT chiếm 70% và điểm từ khóa chiếm 30% ($0.7 \times \text{bert\_scores} + 0.3 \times \text{keyword\_scores}$). Sự kết hợp này tận dụng khả năng hiểu ngữ nghĩa sâu của BERT và tính đặc trưng rõ ràng của từ khóa, giúp kết quả chính xác và đáng tin cậy hơn.
   - **Xác định Độ tin cậy:** Cuối cùng, thuật toán chọn thể loại con có điểm số tổng hợp cao nhất. Ngoài ra, nó cũng có một ngưỡng (< 0.3) để xác định xem mô hình có đủ tự tin để phân loại không. Nếu điểm số quá thấp, nó sẽ gán nhãn '*spam_khac*', giúp tránh việc phân loại sai các tin nhắn mơ hồ.

Tóm lại, phương pháp này là một ví dụ xuất sắc về cách sử dụng học bán giám sát để giải quyết một bài toán phức tạp. Nó kết hợp một mô hình ngôn ngữ lớn (BERT) với một kỹ thuật đơn giản nhưng hiệu quả (đối sánh từ khóa) để phân loại spam một cách thông minh và linh hoạt, ngay cả khi không có đủ dữ liệu đã được gán nhãn.

1.20 Thực thi phương pháp

Đoạn code dưới đây thực thi phương pháp mà nhóm đã đề xuất thực hiện.

```
def classify_spam_subcategory(spam_texts, model, tokenizer, device,idf):
    """Combine BERT embeddings with keyword matching"""
    if not spam_texts:
        return []

    # 1. BERT embeddings
    spam_embeddings = get_embeddings_tfidf(spam_texts, model, tokenizer, device, idf)

    # 2. Category reference embeddings
    reference_texts = {
        'spam_quangcao': "promotional discount sale offer prize win money gift free",
        'spam_hethong': "security account system alert notification verify login password"
    }

    reference_embeddings = {}
    for category, ref_text in reference_texts.items():
        ref_emb = get_embeddings_tfidf([ref_text], model, tokenizer, device, idf)[0]
        reference_embeddings[category] = ref_emb

    subcategories = []

    for i, (text, text_embedding) in enumerate(zip(spam_texts, spam_embeddings)):
        # A. BERT semantic similarity
        bert_scores = {}
```

```
        for category, ref_emb in reference_embeddings.items():
            similarity = np.dot(text_embedding, ref_emb) / (
                np.linalg.norm(text_embedding) * np.linalg.norm(ref_emb)
            )
            bert_scores[category] = similarity

        # B. Keyword matching (existing logic)
        keyword_scores = {}
        text_lower = text.lower()
        category_keywords = {
        'spam_quangcao': [
            # Vietnamese advertising keywords
            'ếkhuyn mãi', 'ảgim giá', 'sale', 'ưu đãi', 'mua ngay', 'giá ẻr', 'ễmin phí',
            'quà ặtng', 'voucher', 'coupon', 'ảgii ưởthng', 'trúng ưởthng', 'ơc ộhi', 'trúng
    ',
            # English advertising keywords
            'discount', 'sale', 'offer', 'promotion', 'free', 'deal', 'buy now', 'limited
    time',
            'special offer', 'bargain', 'cheap', 'save money', 'win', 'prize', 'gift', 'won
    ',
            'congratulations', 'claim', 'click here', '$', 'money', 'cash'
        ],
        'spam_hethong': [
            # Vietnamese system keywords
            'thông báo', 'ảcnh báo', 'tài ảkhon', 'ảbo ậmt', 'xác ậnhn', 'ậcp ậnht',
            'ệh ốthng', 'đăng ậnhp', 'ậmt ẩkhu', 'ịb khóa', 'ếht ạhn', 'gia ạhn', 'khóa',
            # English system keywords
            'notification', 'alert', 'account', 'security', 'confirm', 'update',
            'system', 'login', 'password', 'locked', 'expired', 'renewal', 'verify',
            'suspended', 'warning', 'breach', 'urgent', 'immediately'
        ]
        }

        for category, keywords in category_keywords.items():
            score = sum(1 for keyword in keywords if keyword in text_lower)
            keyword_scores[category] = score / len(keywords)  # Normalize

        # C. Combine scores (weighted)
        final_scores = {}
        for category in bert_scores.keys():
            # 70% BERT, 30% keywords
            final_scores[category] = 0.7 * bert_scores[category] + 0.3 * keyword_scores[
    category]

        # D. Choose best category
        if max(final_scores.values()) < 0.3:  # Low confidence
            best_category = 'spam_khac'
        else:
            best_category = max(final_scores, key=final_scores.get)

        subcategories.append(best_category)

        print(f"Text: {text[:50]}...")
        print(f"BERT scores: {bert_scores}")
        print(f"Keyword scores: {keyword_scores}")
        print(f"Final: {best_category}")

    return subcategories
```

II. Streamlit