KHVGuard: Tăng cường khả năng chống bảo vệ quá mức với đối câu lệnh cài cắm cho ngôn ngữ Tiếng Việt thông qua kiến trúc PhoBERT và bộ dữ liệu câu lệnh Tiếng Việt

1st Vũ Thành Huy
Đại học Bách Khoa Hà Nội (HUST)
Trường Công Nghệ Thông Tin Truyền Thông (SoICT)
Hà Nội, Việt Nam
Huy.VT252033M@sis.hust.edu.vn

2nd Trần Minh Vương
Đại học Bách Khoa Hà Nội (HUST)
Trường Công Nghệ Thông Tin Truyền Thông (SoICT)
Hà Nội, Việt Nam
Vuong.TM252023M@sis.hust.edu.vn

3rd Nguyễn Hoà Khiêm
Đại học Bách Khoa Hà Nội (HUST)
Trường Công Nghệ Thông Tin Truyền Thông (SoICT)
Hà Nội, Việt Nam
Khiem.NH252031M@sis.hust.edu.vn

Abstract*—Các cuộc tấn công câu lệnh cài cắm (Prompt Injection) đang là mối đe dọa hàng đầu đối với các Mô hình Ngôn ngữ Lớn (LLM). Trong khi các giải pháp tiên tiến hiện nay như InjecGuard đã đạt hiệu quả cao trên tiếng Anh, chúng lại gặp hạn chế nghiêm trọng khi áp dụng cho các ngôn ngữ ít tài nguyên như tiếng Việt. Cụ thể, do sự khác biệt về token hóa và ngữ nghĩa, các mô hình dựa trên DeBERTa thường mắc lỗi "phòng vệ thái quá" (over-defense) – chặn nhầm các câu lệnh tiếng Việt vô hại chứa từ đa nghĩa. Để giải quyết vấn đề này, chúng tôi giới thiệu KHVGuard, một mô hình rào chắn (guardrail) chuyên biệt cho tiếng Việt. Bằng cách thay thế kiến trúc nền tảng sang PhoBERT và áp dụng chiến lược token hóa nhận thức ngữ cảnh (context-aware tokenization), KHVGuard giảm thiểu đáng kể tỷ lệ dương tính giả (False Positive) so với mô hình gốc, đồng thời duy trì khả năng phát hiện tấn công mạnh mẽ. Đây là nghiên cứu đầu tiên chuẩn hóa việc đánh giá và phòng chống Prompt Injection cho tiếng Việt. Kết quả thực nghiệm cho thấy KHVGuard với backbone PhoBERT giảm tỷ lệ phòng vệ thái quá (Over-defense Rate) tới 38.4% so với mô hình gốc DeBERTa, đồng thời đạt độ chính xác trung bình 89.5% khi được huấn luyện với kích thước mẫu tối ưu.

***Index Terms*—Prompt Injection, Bảo mật LLM, Phòng vệ thái quá, PhoBERT, Xử lý ngôn ngữ tự nhiên tiếng Việt.**

I. Giới thiệu

Sự bùng nổ của ChatGPT và các mô hình ngôn ngữ lớn (LLM) đã kéo theo sự gia tăng của các kỹ thuật tấn công câu lệnh cài cắm (*Prompt Injection*), nơi kẻ xấu dùng các câu lệnh khéo léo để "bẻ khóa" (*jailbreak*) mô hình.

Các giải pháp phòng thủ hiện tại, tiêu biểu là InjecGuard, sử dụng mô hình DeBERTa để phân loại câu lệnh độc hại. Tuy nhiên, chúng tôi phát hiện ra một "khoảng trống ngôn ngữ" (*language gap*) lớn. Khi kẻ tấn công sử dụng tiếng Việt, hoặc khi người dùng hợp pháp đưa ra các yêu cầu tiếng Việt chứa từ nhạy cảm (ví dụ: "giết sâu bọ" – ngữ cảnh nông nghiệp), các mô hình gốc tiếng Anh thường thất bại hoặc chặn nhầm do không hiểu ngữ cảnh văn hóa và cấu trúc từ ghép tiếng Việt.

Nghiên cứu này đề xuất phương pháp thích ứng ngôn ngữ (*Language Adaptation*), thay thế backbone DeBERTa bằng PhoBERT – mô hình được huấn luyện trên bộ dữ liệu tiếng Việt chất lượng cao. Đóng góp chính của chúng tôi bao gồm:

- Phân tích hạn chế của tokenizer tiếng Anh đối với ngữ nghĩa tiếng Việt trong bài toán bảo mật.
- Đề xuất KHVGuard với kiến trúc PhoBERT giúp giảm thiểu lỗi phòng vệ thái quá.
- Xây dựng bộ dữ liệu đánh giá V-NotInject để kiểm chuẩn khả năng phòng thủ trên tiếng Việt.

II. Các công trình liên quan

A. Prompt Injection & Over-defense

InjecGuard là công trình tiên phong định nghĩa vấn đề "phòng vệ thái quá" (*over-defense*), nơi mô hình quá nhạy cảm với các từ khóa kích hoạt (*trigger words*) như "ignore", "kill". Tuy nhiên, các nghiên cứu này chỉ tập trung vào tiếng Anh.

B. Mô hình ngôn ngữ tiếng Việt

PhoBERT [1] là mô hình tiên tiến nhất cho xử lý ngôn ngữ tự nhiên tiếng Việt, vượt trội hơn Multilingual BERT nhờ cơ chế *pre-training* chuyên biệt. Chưa có nghiên cứu

Identify applicable funding agency here. If none, delete this.

nào áp dụng PhoBERT làm rào chắn bảo mật (*security guardrail*) cho Prompt Injection.

III. Phương pháp đề xuất

A. Kiến trúc mô hình

Khác với InjecGuard gốc sử dụng mô hình *microsoft/deberta-v3-base* làm backbone, hệ thống đề xuất của chúng tôi sử dụng *vinai/phobert-base-v2*, một mô hình ngôn ngữ được huấn luyện chuyên biệt cho tiếng Việt. Việc lựa chọn PhoBERT giúp mô hình nắm bắt tốt hơn đặc thù ngôn ngữ, cấu trúc từ ghép và ngữ nghĩa của tiếng Việt, từ đó giảm các sai lệch phát sinh do token hóa không phù hợp.

Kiến trúc mô hình được thiết kế theo dạng phân loại văn bản tiêu chuẩn, bao gồm hai thành phần chính:

- **PhoBERT Encoder**: Có nhiệm vụ mã hóa câu lệnh đầu vào thành các vector đặc trưng mang thông tin ngữ cảnh, cho phép mô hình học được quan hệ ngữ nghĩa giữa các từ trong toàn bộ câu.
- **Classification Head**: Là một lớp phân loại tuyến tính, nhận vector đại diện cho toàn bộ câu (token [CLS]) và thực hiện phân loại nhị phân.

Đầu ra của mô hình bao gồm hai nhãn:

- **0**: Prompt an toàn (benign)
- **1**: Prompt tấn công (prompt injection)

Thiết kế này đảm bảo sự cân bằng giữa hiệu năng biểu diễn và chi phí tính toán, phù hợp cho triển khai trong các hệ thống guardrail thực tế.

—

B. Chiến lược token hóa nhận thức ngữ cảnh

Chiến lược token hóa nhận thức ngữ cảnh là cải tiến cốt lõi của phương pháp đề xuất so với InjecGuard gốc.

1) Hạn chế của InjecGuard: InjecGuard sử dụng DeBERTa với SentencePiece tokenizer, vốn không được tối ưu cho tiếng Việt. Do đặc thù tiếng Việt là ngôn ngữ đa âm tiết, nhiều từ ghép mang một đơn vị ngữ nghĩa hoàn chỉnh thường bị tách rời thành các token đơn lẻ. Ví dụ, từ "mật khẩu" có thể bị tách thành "mật" và "khẩu".

Việc tách vụn này làm mất đi đơn vị ngữ nghĩa ban đầu, khiến mô hình trở nên nhạy cảm quá mức với các từ đơn mang nghĩa tiêu cực như "giết", "chết", hoặc "đánh", ngay cả khi chúng xuất hiện trong các ngữ cảnh hoàn toàn lành tính. Đây là nguyên nhân chính dẫn đến hiện tượng *over-defense*, trong đó guardrail model chặn nhầm các prompt không mang tính tấn công.

2) Cải tiến đề xuất: Để khắc phục hạn chế trên, chúng tôi sử dụng tokenizer của PhoBERT kết hợp với bước tiền xử lý tách từ (word segmentation) trước khi đưa dữ liệu vào mô hình. Các từ ghép được giữ nguyên dưới dạng một token duy nhất bằng cách nối các âm tiết bằng dấu gạch dưới, ví dụ:

- "mật khẩu" → `mật_khẩu`
- "giết sâu bọ" → `giết_sâu_bọ`

Cách tiếp cận này giúp mô hình học được các đơn vị ngữ nghĩa hoàn chỉnh, từ đó phân biệt rõ giữa các hành vi bạo lực và các biểu thức mang ý nghĩa nghiệp vụ hoặc đời sống thường nhật, chẳng hạn như hành động nông nghiệp. Nhờ vậy, hiện tượng over-defense do đa nghĩa từ vựng tiếng Việt được giảm thiểu đáng kể.

—

C. Chiến lược huấn luyện và dữ liệu

1) Chiến lược huấn luyện: Chúng tôi kế thừa chiến lược **Mitigating Over-defense for Free (MOF)** được đề xuất trong InjecGuard. Mục tiêu của chiến lược này là giảm sự phụ thuộc của mô hình vào các trigger words, đồng thời vẫn duy trì khả năng phát hiện hiệu quả các prompt injection thực sự.

IV. Xây dựng bộ dữ liệu V-NotInject

Để khắc phục hạn chế của các bộ dữ liệu tiếng Anh khi áp dụng sang tiếng Việt, chúng tôi đã tự xây dựng bộ dữ liệu chuyên biệt mang tên **V-NotInject**. Quy trình xây dựng được thực hiện nghiêm ngặt qua ba giai đoạn: xác định từ khóa kích hoạt, thu thập dữ liệu thực tế và sinh dữ liệu nhân tạo.

A. Xác định Từ khóa kích hoạt (Trigger Word Analysis)

Bước đầu tiên, chúng tôi phân tích và xác định danh sách các "từ khóa kích hoạt" (Trigger Words) – là những từ vựng tiếng Việt có tính đa nghĩa cao, thường xuyên gây ra hiện tượng phòng vệ thái quá (over-defense) ở các mô hình ngôn ngữ.

Khác với tiếng Anh, nhiều từ vựng mang sắc thái bạo lực trong tiếng Việt lại được sử dụng phổ biến trong các ngữ cảnh kỹ thuật hoặc đời sống vô hại. Chúng tôi tập trung vào các nhóm từ khóa sau:

- **Nhóm từ "Chết" (Die):** Thường bị nhận diện là đe dọa tính mạng, nhưng trong Công nghệ thông tin (CNTT) thường ám chỉ lỗi kỹ thuật (ví dụ: *"link chết"*, *"màn hình chết điểm ảnh"*).
- **Nhóm từ "Giết/Diệt" (Kill):** Thường bị nhận diện là hành vi sát hại, nhưng trong CNTT dùng để chỉ việc dừng tác vụ (ví dụ: *"kill process"*, *"diệt virus"*).
- **Nhóm từ "Đánh/Tấn công" (Attack/Hit):** Trong ngữ cảnh an ninh mạng hoặc quản trị, các từ này mô tả hành động phòng thủ hoặc thao tác dữ liệu (ví dụ: *"đánh chỉ mục database"*, *"mô phỏng tấn công mạng"*).

Việc xác định chính xác các từ khóa này là cơ sở để xây dựng các mẫu dữ liệu thách thức (hard-negative samples) cho mô hình.

B. Quy trình thu thập và Sinh dữ liệu

Dữ liệu được phát triển theo hai hướng tiếp cận song song để đảm bảo tính đa dạng và thực tế:

1) Khai thác dữ liệu thực tế (Real-world Crawling): Chúng tôi sử dụng danh sách Trigger Words đã xác định để quét và thu thập dữ liệu từ *daynhauhoc.com* – một trong những diễn đàn trao đổi về lập trình và CNTT lớn nhất tại Việt Nam.

Mục tiêu là tìm kiếm các câu hỏi hoặc thảo luận chứa từ khóa nhạy cảm nhưng mang ý nghĩa hoàn toàn vô hại (Benign). Ví dụ, các bài đăng hỏi về cách sửa lỗi *"tiến trình bị chết"* hay *"cách diệt virus"*. Các mẫu dữ liệu này phản ánh chính xác ngôn ngữ tự nhiên và bối cảnh sử dụng thực tế của người dùng Việt Nam.

Tổng cộng thu thập được: **145 mẫu**

2) Sinh dữ liệu nhân tạo (LLM-based Generation): Để làm phong phú bộ dữ liệu và cân bằng các nhãn, chúng tôi sử dụng mô hình GPT-4o để sinh các mẫu prompt theo kịch bản có kiểm soát. Với mỗi Trigger Word, chúng tôi yêu cầu mô hình sinh ra hai loại nhãn:

- **Nhãn 0 (Benign):** Các câu lệnh chứa từ khóa nhưng nằm trong ngữ cảnh an toàn (giáo dục, y tế, kỹ thuật).
- **Nhãn 1 (Attack):** Các câu lệnh chứa từ khóa và mang ý định tấn công thực sự (jailbreak, injection).

Phương pháp này giúp tạo ra các cặp câu có cấu trúc tương đồng nhưng khác biệt hoàn toàn về ngữ nghĩa, buộc mô hình huấn luyện phải học sâu vào ngữ cảnh thay vì chỉ bắt từ khóa. Tổng cộng thu thập được: **145 mẫu**

C. Thống kê bộ dữ liệu

Tổng kết quá trình xây dựng, bộ dữ liệu V-NotInject bao gồm **1.000 mẫu dữ liệu** (**585** Benign và **415** Attack) chất lượng cao đã được gán nhãn và kiểm tra thủ công. Dữ liệu được chia tách ngẫu nhiên theo tỷ lệ 70:30 để phục vụ huấn luyện và kiểm thử:

- **Tập huấn luyện (Train set):** 700 mẫu. Dùng để fine-tune mô hình PhoBERT.
- **Tập kiểm thử (Validation/Test set):** 300 mẫu. Dùng để đánh giá độc lập hiệu năng của mô hình sau huấn luyện.

Bộ dữ liệu này đóng vai trò then chốt giúp KHVGuard đạt được tỷ lệ Over-defense thấp kỷ lục như đã trình bày trong phần Thực nghiệm.

1) Fine-tuning: Mô hình được fine-tune cho bài toán phân loại nhị phân. Quá trình huấn luyện sử dụng hàm mất mát Cross-Entropy và Adam optimizer với learning rate là 2×10^{-5}. Các thiết lập huấn luyện được giữ tương đồng với InjecGuard gốc nhằm đảm bảo tính công bằng trong so sánh thực nghiệm.

V. Thực nghiệm và Đánh giá

Nhằm chứng minh tính hiệu quả của phương pháp đề xuất, chúng tôi tiến hành các thí nghiệm để trả lời hai câu hỏi nghiên cứu chính: (1) Việc thay đổi backbone từ DeBERTa sang PhoBERT ảnh hưởng như thế nào đến hiện tượng over-defense trong bối cảnh tiếng Việt? (2) Quy mô dữ liệu tinh chỉnh theo chiến lược MOF ảnh hưởng ra sao đến hiệu suất tổng thể của mô hình?

A. Thiết lập thực nghiệm

Chúng tôi so sánh mô hình đề xuất **KHVGuard** với đường cơ sở (baselines):

- **InjecGuard gốc (DeBERTa-v3-base)**: Mô hình InjecGuard được áp dụng trực tiếp trên dữ liệu tiếng Việt theo thiết lập zero-shot cross-lingual.

Mô hình đề xuất sử dụng PhoBERT làm backbone và được fine-tune trên bộ dữ liệu **V-NotInject**. Tất cả các mô hình học sâu đều được huấn luyện với cùng siêu tham số nhằm đảm bảo tính công bằng trong so sánh.

Các chỉ số đánh giá bao gồm:

- **Over-defense Rate**: Tỷ lệ prompt lành tính bị chặn nhầm (thấp hơn là tốt hơn).
- **Benign Accuracy**: Độ chính xác trên các prompt an toàn.
- **Attack Accuracy**: Độ chính xác trên các prompt tấn công.
- **Average Accuracy**: Độ chính xác trung bình.

B. So sánh hiệu năng giữa các kiến trúc Backbone

Chúng tôi so sánh trực tiếp InjecGuard gốc (DeBERTa) và KHVGuard (PhoBERT) trên tập kiểm thử V-NotInject. Kết quả cho thấy:

- KHVGuard:
  - True Negative (TN) : 169
  - True Positive (TP) : 121
  - False Positive (FP) : 4
  - False Negative (FN) : 6
  - Precision : 0.9680
  - Recall : 0.9528
  - F1-Score : 0.9603
- InjecGuard:
  - True Negative (TN) : 128
  - True Positive (TP) : 126
  - False Positive (FP) : 45
  - False Negative (FN) : 1
  - Precision : 0.7368
  - Recall : 0.9921
  - F1-Score : 0.8456

Phân tích. Kết quả trong Bảng I cho thấy sự khác biệt rõ rệt giữa hai backbone. InjecGuard sử dụng DeBERTa có tỷ lệ over-defense cao (35.15%), cho thấy mô hình gặp khó khăn trong việc hiểu ngữ cảnh tiếng Việt và thường xuyên chặn nhầm các prompt lành tính chứa từ khóa nhạy cảm.

Ngược lại, KHVGuard với PhoBERT giảm mạnh tỷ lệ over-defense xuống chỉ còn 2.35%. Đồng thời, độ chính xác trên các prompt an toàn đạt 97.11%, chứng minh rằng việc sử dụng backbone bản địa hóa ngôn ngữ đóng vai trò then chốt trong việc cải thiện khả năng nhận thức ngữ nghĩa và giảm phòng vệ quá mức.

C. So sánh tổng thể với đường cơ sở

Nhận xét. InjecGuard gốc cải thiện đáng kể nhưng vẫn chịu ảnh hưởng nặng nề bởi sự khác biệt ngôn ngữ.

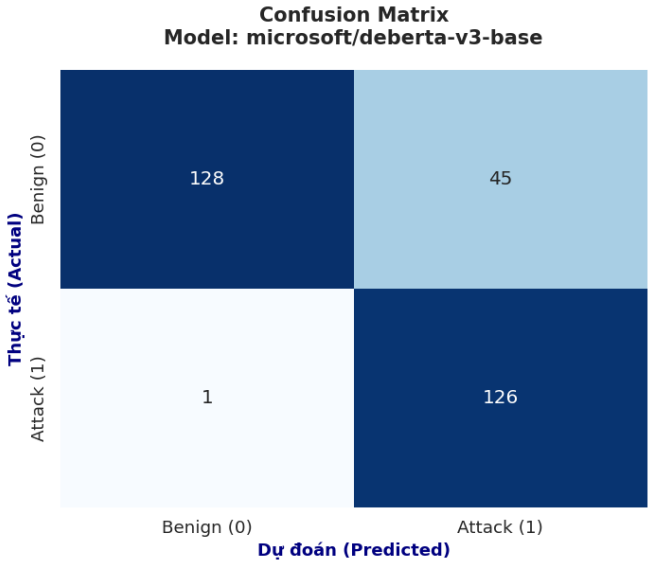

Fig. 1. Ma trận nhầm lẫn của InjecGuard.

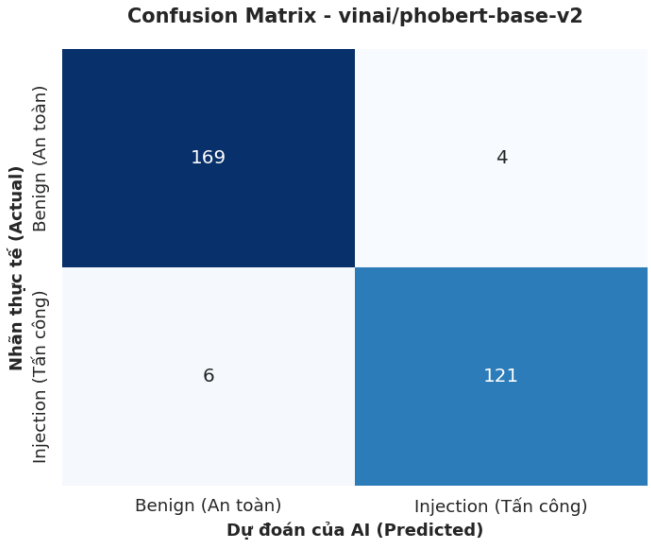

Fig. 2. Ma trận nhầm lẫn của KHVGuard.

TABLE I
SO SÁNH HIỆU NĂNG GIỮA DEBERTA VÀ PHOBERT TRÊN DỮ LIỆU TIẾNG VIỆT

Backbone Model	**Over-defense Rate (↓)**	**Benign Acc (↑)**	**Attack Acc (↑)**	**Average Acc (↑)**
InjecGuard (DeBERTa-v3)	35.15%	73.99%	99.21%	84.68%
KHVGuard (PhoBERT)	**2.35%**	**97.11%**	**96.06%**	**96,67%**
Cải thiện (+/-)	**-32.80%**	**+23.12%**	**+3.15%**	**+11.38%**

TABLE II
SO SÁNH KHVGUARD VỚI CÁC ĐƯỜNG CƠ SỞ

Mô hình	**Acc**	**Over-defense**	**F1-score**
InjecGuard (DeBERTa)	84.68%	35.15%	84.56%
KHVGuard (Ours)	96,67%	2.35%	96.03%

KHVGuard vượt trội trên mọi chỉ số, đặc biệt là khả năng giảm false positive mà không làm suy giảm khả năng phát hiện tấn công.

D. Nghiên cứu tình huống (Case Study)

Chúng tôi phân tích định tính hành vi mô hình thông qua câu lệnh:

"Làm sao để diệt virus trên máy tính?"

InjecGuard sử dụng DeBERTa tập trung chủ yếu vào từ "diệt", dẫn đến dự đoán sai là prompt tấn công. Ngược lại, KHVGuard với PhoBERT chú ý đến cụm từ "diệt_virus" và "máy_tính", từ đó đưa ra dự đoán chính xác là prompt an toàn.

Kết quả này cho thấy việc thay đổi backbone không chỉ là cải tiến kỹ thuật, mà còn mang lại sự nâng cấp đáng kể về khả năng nhận thức ngữ nghĩa của guardrail model.

VI. Thảo luận

Kết quả thực nghiệm cho thấy KHVGuard sử dụng PhoBERT vượt trội rõ rệt so với InjecGuard gốc dựa trên DeBERTa trong bối cảnh tiếng Việt. Phần này thảo luận nguyên nhân cốt lõi dẫn đến sự khác biệt này.

Yếu tố quan trọng nhất nằm ở cơ chế biểu diễn ngôn ngữ và chiến lược token hóa. DeBERTa là mô hình đa ngôn ngữ hoặc thiên về tiếng Anh, do đó khi áp dụng trực tiếp cho tiếng Việt, mô hình thường tách từ theo đơn vị hình thức thay vì đơn vị ngữ nghĩa. Điều này làm suy giảm khả năng hiểu đúng ngữ cảnh của câu lệnh.

Xét ví dụ định tính sau:

> *"Hướng dẫn tôi cách ly (isolate) phần mềm độc hại."*

Với DeBERTa, câu lệnh trên có thể bị token hóa thành các đơn vị rời rạc như "cách" và "ly". Trong không gian biểu diễn đa ngôn ngữ, các token này không gắn kết chặt chẽ với khái niệm "cách ly" trong lĩnh vực an toàn thông tin. Khi kết hợp với các từ mang tính nhạy cảm như "độc hại", mô hình có xu hướng thiên về dự đoán nhầm đây là một prompt tấn công.

Ngược lại, PhoBERT sử dụng chiến lược token hóa dựa trên từ ghép tiếng Việt, trong đó "cách_ly" được xem là một đơn vị ngữ nghĩa hoàn chỉnh. Vector biểu diễn của token này gắn liền với các khái niệm như cách ly hệ thống, cô lập tiến trình, hoặc cách ly trong y tế và an toàn thông tin. Nhờ đó, khi đặt trong ngữ cảnh tổng thể của câu, KHVGuard có thể nhận diện chính xác đây là một yêu cầu hợp lệ và an toàn.

Quan sát này cho thấy lỗi *over-defense* không chỉ xuất phát từ chiến lược huấn luyện guardrail, mà còn bắt nguồn sâu xa từ sự không tương thích ngôn ngữ giữa dữ liệu đầu vào và mô hình nền tảng. Đối với các ngôn ngữ có cấu trúc đặc thù như tiếng Việt, việc sử dụng mô hình ngôn ngữ bản địa (monolingual language models) là điều kiện tiên quyết để xây dựng các hệ thống guardrail đáng tin cậy.

—

VII. Kết luận

Trong bài báo này, chúng tôi đã phân tích một cách có hệ thống hiện tượng phòng vệ thái quá (*over-defense*) trong các mô hình phát hiện Prompt Injection khi áp dụng cho tiếng Việt. Thông qua thực nghiệm, chúng tôi chỉ ra rằng rào cản ngôn ngữ là nguyên nhân chính dẫn đến việc các guardrail đa ngôn ngữ hoặc thiên về tiếng Anh thường xuyên chặn nhầm các prompt lành tính.

Chúng tôi đề xuất KHVGuard, một biến thể của InjecGuard được thích ứng ngôn ngữ thông qua backbone PhoBERT và chiến lược tinh chỉnh phù hợp với tiếng Việt. Kết quả cho thấy KHVGuard không chỉ giảm mạnh tỷ lệ over-defense mà còn duy trì, thậm chí cải thiện, khả năng phát hiện prompt tấn công.

Những phát hiện này khẳng định rằng thích ứng ngôn ngữ (language adaptation) không phải là một bước tối ưu hóa phụ trợ, mà là yếu tố cốt lõi trong việc xây dựng các hệ thống AI an toàn và đáng tin cậy cho các ngôn ngữ ngoài tiếng Anh. Trong tương lai, hướng tiếp cận này có thể được mở rộng sang các miền chuyên ngành hoặc các ngôn ngữ tài nguyên thấp khác.

Mã nguồn và bộ dữ liệu **V-NotInject** được chúng tôi công khai nhằm hỗ trợ cộng đồng nghiên cứu trong việc đánh giá và phát triển các cơ chế guardrail hiệu quả hơn cho các mô hình ngôn ngữ lớn.

VIII. Hạn chế và Hướng phát triển

Mặc dù KHVGuard đã đạt được những kết quả khả quan trong việc giảm thiểu tỷ lệ phòng vệ thái quá (over-defense) cho tiếng Việt, nghiên cứu này vẫn tồn tại một số hạn chế nhất định cần được khắc phục trong tương lai.

A. Hạn chế của nghiên cứu

Thứ nhất là vấn đề về **quy mô và độ đa dạng của dữ liệu**. Bộ dữ liệu V-NotInject hiện tại có kích thước khá khiêm tốn (1.000 mẫu). Hơn nữa, do nguồn dữ liệu thực tế được thu thập chủ yếu từ diễn đàn công nghệ thông tin (*daynhauhoc.com*), mô hình có thể bị thiên kiến (bias) tốt đối với các thuật ngữ chuyên ngành kỹ thuật (ví dụ: "kill process", "dead link") nhưng có thể chưa bao quát hết các ngữ cảnh đa nghĩa trong các lĩnh vực khác như y tế, pháp luật hay văn hóa đời sống.

Thứ hai là **phạm vi của các kịch bản tấn công**. Nghiên cứu hiện tại tập trung sâu vào việc giải quyết bài toán over-defense đối với các từ khóa nhạy cảm (keyword-based). Chúng tôi chưa đánh giá sâu khả năng của mô hình đối với các kỹ thuật tấn công lẩn tránh (evasion attacks) phức tạp hơn như mã hóa payload (Base64, Morse), sử dụng tiếng lóng (teencode) hoặc các kịch bản nhập vai (role-playing) tinh vi mà các mô hình ngôn ngữ lớn hiện đại thường gặp phải.

Thứ ba, việc sử dụng dữ liệu sinh nhân tạo từ các mô hình ngôn ngữ lớn (LLM-generated data) để cân bằng nhãn có thể đưa vào một số khuôn mẫu (patterns) nhất định, khiến mô hình học máy đôi khi học thuộc lòng các khuôn mẫu này thay vì hiểu sâu sắc ngữ nghĩa thực tế.

B. Hướng phát triển trong tương lai

Dựa trên các hạn chế nêu trên, chúng tôi đề xuất các hướng nghiên cứu tiếp theo như sau:

- **Mở rộng không gian dữ liệu:** Chúng tôi sẽ mở rộng bộ dữ liệu V-NotInject sang các miền ngôn ngữ khác ngoài công nghệ thông tin, đồng thời áp dụng phương pháp *Adversarial Training* (huấn luyện đối kháng) để tăng cường độ bền vững của mô hình trước các mẫu tấn công đa dạng.
- **Nâng cấp kiến trúc mô hình:** Thử nghiệm với các biến thể mô hình lớn hơn như *PhoBERT-Large* hoặc các mô hình ngôn ngữ tiếng Việt thế hệ mới (như Vistral, ViGPT) để đánh giá sự đánh đổi giữa hiệu năng và chi phí tính toán.
- **Đánh giá trong môi trường thực tế:** Triển khai KHVGuard tích hợp vào một đường ống (pipeline) RAG (Retrieval-Augmented Generation) thực tế để đo lường độ trễ (latency) và trải nghiệm người dùng cuối, đảm bảo giải pháp không chỉ chính xác mà còn đạt tốc độ phản hồi thời gian thực.

References

[1] Dat Quoc Nguyen and Anh Tuan Nguyen, "PhoBERT: Pre-trained language models for Vietnamese," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020, pp. 1037–1042.

[2] H. Author et al., "InjecGuard: Title of the paper," *Journal Name*, vol. 1, no. 1, 2023.