SafeConv: A Benchmark and Model for Safe Open-domain Conversation

Nguyen Lu Bao Khang[1], Tran Pham Hoang phong[2], Nguyen Van Phong[3], Le Minh Thuan[4]
[1]22DH114577@st.huflit.edu.vn [2]22DH114682@st.huflit.edu.vn
[3]22DH112745@st.huflit.edu.vn [4]22DH113570@st.huflit.edu.vn

1 Giới Thiệu

Một trong những thách thức lớn trong phát triển chatbot hiện nay là đảm bảo hệ thống có thể giao tiếp một cách an toàn và thân thiện với người dùng, tránh đưa ra các phản hồi tiêu cực, gợi ý nguy hiểm hoặc đồng tình với các quan điểm lệch lạc. Tuy nhiên, phần lớn các bộ dữ liệu đối thoại hiện có chưa cung cấp đầy đủ chú thích cần thiết để phát hiện và điều chỉnh các hành vi không an toàn này.

Nhằm giải quyết vấn đề trên, nhóm tác giả Zhang et al. đã xây dựng bộ dữ liệu mới có tên **SafeConv** phục vụ nghiên cứu về an toàn hội thoại. SafeConv không chỉ cung cấp nhãn an toàn ở cấp độ phát ngôn (*utterance-level safety labels*), mà còn chú thích các cụm từ gây nguy hiểm (*unsafe spans*) và đưa ra các phản hồi thay thế an toàn (*safe alternative responses*) nhằm duy trì mạch hội thoại một cách tự nhiên và nhất quán.

Dựa trên các chú thích toàn diện này, nhóm tác giả đề xuất ba mô hình xử lý: **Checker** để phát hiện phát ngôn không an toàn, **Tagger** để trích xuất các cụm từ nguy hiểm, và **Rewriter** để chuyển đổi phản hồi thành phiên bản an toàn hơn. Trong nghiên cứu này, chúng tôi triển khai lại framework SafeConv trong bối cảnh đa ngôn ngữ, nhằm khảo sát tính khả thi khi áp dụng vào các ngôn ngữ khác ngoài tiếng Anh, cụ thể là tiếng Việt.

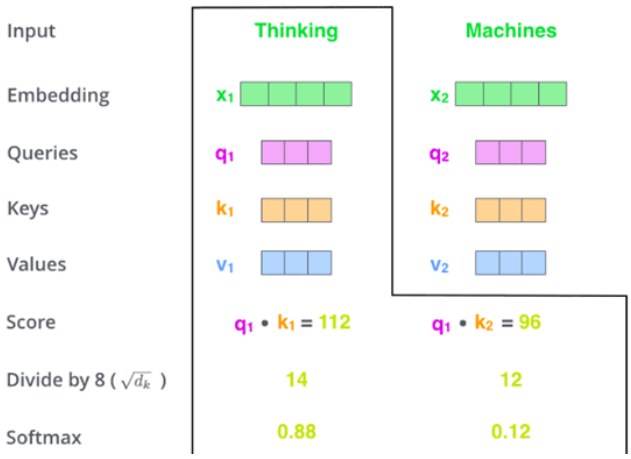
2 Cơ sở lý thuyết

2.1 Self-Attention

Mục tiêu: Cơ chế self-attention cho phép mô hình so sánh từng cặp từ trong chuỗi đầu vào, bao gồm cả chính nó, để xác định mức độ quan trọng (weight) mà mô hình cần chú ý tới. Điều này giúp mô hình hiểu rõ ý nghĩa của từng từ trong ngữ cảnh cụ thể, thay vì chỉ dựa vào ý nghĩa tổng quát khi từ đó đứng một mình.

Cơ chế hoạt động: Với mỗi chuỗi đầu vào đã được nhúng (embedding), mỗi token sẽ được biến đổi thành ba vectơ riêng biệt:

- **Query (Q):** Đại diện cho từ đang được xét.
- **Key (K):** Đại diện cho các từ xung quanh.
- **Value (V):** Thông tin cuối cùng cần được tổng hợp lại.



Hình 1: Minh họa cơ chế hoạt động Self-Attention (nguồn:Viblo)

Quá trình tính toán như sau:

1. Tính tích vô hướng giữa "Q" và "K" để đo mức độ liên quan giữa từ đang xét với các từ còn lại.
2. Áp dụng SoftMax để chuẩn hóa thành các trọng số chú ý (Attention Weights).
3. 1. Nhân các trọng số trên với "**V**" để tạo ra biểu diễn ngữ cảnh mới cho từng từ ta đang xét.

Biểu diễn toán học của Self-Attention
Công thức:

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

Trong đó:

- d_k: là kích thước chiều của vectơ Key.
- T: là ma trận chuyển vị.

Ưu điểm:

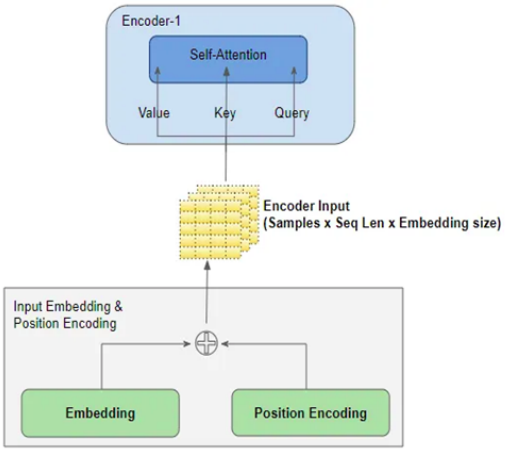
- Cho phép nắm bắt ngữ cảnh toàn cục (global context).
- Học được các mối quan hệ xa – gần trong câu mà không bị giới hạn độ dài.
- Có thể xử lý song song tất cả từ trong chuỗi (khác RNN).

Nhược điểm:

- Cần tính toán ma trận lớn (chi phí O(n²) với n là số từ), khó mở rộng cho chuỗi quá dài.
- Thiếu thông tin vị trí (position) nếu không có kỹ thuật positional encoding.

2.2 Multi-Head Attention

Mục tiêu:
Cơ chế *Multi-Head Attention* giúp mô hình tập trung đồng thời vào nhiều khía cạnh khác nhau của ngữ cảnh, thay vì chỉ dùng một hướng chú ý duy nhất. Điều này tăng khả năng nắm bắt các mối quan hệ ngữ nghĩa phức tạp giữa các từ.



Hình 2: Minh họa cơ chế hoạt động Multi-head Sefl-Attention (nguồn:Viblo)

Cách hoạt động:

- Thay vì chỉ tạo một bộ Query (Q), Key (K) và Value (V), thì mô hình sẽ tạo ra n "đầu"(heads) khác nhau.
- Mỗi đầu sẽ có tập trọng số riêng cho Q, K, V (khởi tạo khác nhau), từ đó học được những kiểu tương quan khác nhau trong câu.

Quy trình các bước:

1. Nhân input với các trọng số riêng để tạo Q, K, V cho mỗi head.
2. Mỗi Head tính attention riêng theo công thức:

$$\text{head}_i = \text{Attention}(Q_i, K_i, V_i) = \text{Softmax}\left(\frac{Q_i K_i^\top}{\sqrt{d_k}}\right) V_i \quad (1)$$

3. Các output của từng head được nối lại (concatenate) và đưa qua một lớp tuyến tính (Linear) cuối cùng.

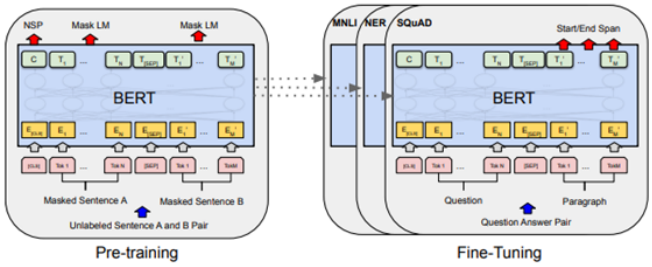
$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_n) W^O$$

Ưu điểm:

- Tăng khả năng học thông tin đa chiều: Mỗi head tập trung vào một mối quan hệ ngữ cảnh khác nhau.
- Học được sâu và chính xác hơn so với chỉ dùng một attention duy nhất.

2.3 BERT

Mô hình kiến trúc: Mô hình kiến trúc: BERT là một mô hình ngôn ngữ sâu được giới thiệu bởi Devlin et al. (2018), mang tính đột phá nhờ khả năng học biểu diễn ngữ nghĩa theo cách hai chiều (bidirectional) thông qua kiến trúc Transformer encoder. Mô hình được huấn luyện theo hai giai đoạn chính:



Hình 3: pre-training và fine-tuning của BERT (nguồn:phamdinhkhanh.github.io)

BERT sử dụng Transformer encoder (Vaswani et al., 2017) với nhiều lớp self-attention để mã hóa toàn bộ chuỗi đầu vào. Mỗi token đầu vào được biểu diễn bằng cách cộng ba vector: **token embedding**, **segment embedding** và **position embedding**.

Chuỗi đầu vào có dạng:

`[CLS]SentenceA[SEP]SentenceB[SEP]`

Trong đó:

- `[CLS]` là token đặc biệt, biểu diễn toàn câu, dùng trong các tác vụ phân loại.
- `[SEP]` là token ngăn cách giữa hai câu.
- Token Type Embedding giúp mô hình phân biệt câu A và B.

Giai đoạn Pre-training (Huấn luyện sơ cấp)

BERT được huấn luyện trước trên tập dữ liệu lớn không gán nhãn thông qua hai tác vụ:

- **Masked Language Modeling (MLM)**: Một số token (15%) được thay thế bằng `[MASK]` và mô hình học cách dự đoán chúng dựa vào ngữ cảnh hai chiều xung quanh.
- **Next Sentence Prediction (NSP)**: Mô hình dự đoán xem câu B có thực sự là câu tiếp theo của câu A hay không. Điều này giúp học quan hệ giữa các câu.

Giai đoạn Fine-tuning (Điều chỉnh cho từng tác vụ):

Sau khi pre-train, BERT được fine-tune trên từng tác vụ cụ thể bằng cách thêm một lớp output phù hợp vào đầu ra của BERT:

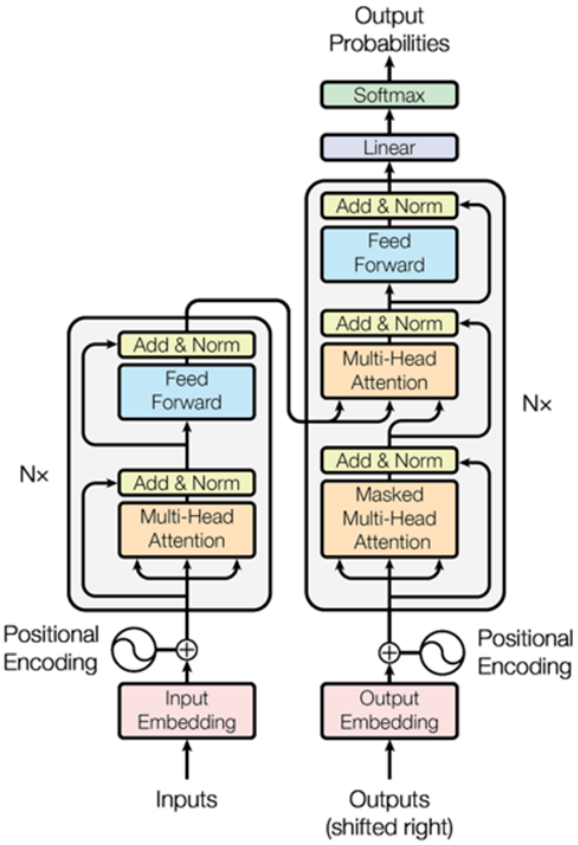
- **Với tác vụ hỏi đáp (VD: SQuAD)**: mô hình dự đoán vị trí bắt đầu và kết thúc câu trả lời trong đoạn văn, tương ứng với hai mũi tên đỏ "Start/End Span".
- **Với phân loại văn bản (VD: MNLI)**: sử dụng vector đầu ra tại `[CLS]` để phân loại.
- **Với gán nhãn thực thể (NER)**: áp dụng softmax trên từng token để dự đoán nhãn.

Quá trình fine-tune rất hiệu quả vì toàn bộ trọng số của mô hình được cập nhật, giúp mô hình thích nghi tốt với nhiệm vụ cụ thể từ dữ liệu nhỏ hơn.

2.4 Transformer

Mô hình kiến trúc: Transformer là một kiến trúc mạng nơ-ron được giới thiệu bởi Vaswani et al. (2017) trong bài báo *"Attention is All You Need"*. Khác với các mô hình tuần tự (RNN, LSTM), Transformer xử lý toàn bộ chuỗi song song thông qua cơ chế attention. Kiến trúc cơ bản bao gồm hai thành phần chính:

- **Encoder:** nhận đầu vào và học biểu diễn ẩn (hidden representations).
- **Decoder:** sinh đầu ra từng bước dựa trên đầu ra của encoder và các từ đã sinh trước đó.



Hình 4: Kiến trúc Transformer (nguồn:viblo)

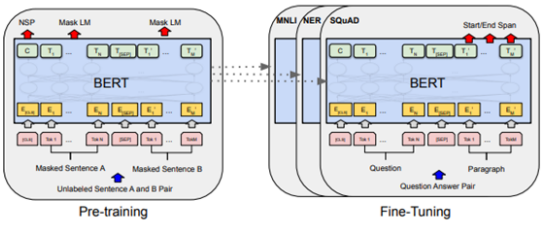
- **Positional Encoding:** Vì Transformer không xử lý dữ liệu theo thứ tự tuần tự như RNN hay LSTM, nên cần thêm thông tin về vị trí từ trong chuỗi. Positional Encoding hoạt động bằng cách sử dụng vector để thể hiện vị trí từ trong câu rồi cộng trực tiếp vào embedding vector.
- **Multi-Head Attention:** Thay vì tính Attention một lần, Transformer tính song

song nhiều "đầu" attention, cho phép mô hình học các mối quan hệ ở nhiều không gian con khác nhau, từ đó giúp mô hình học được nhiều khía cạnh hơn.

- **Kết nối Residual và Layer Normalization:** Kết quả đầu ra của một khối sẽ được cộng với đầu vào ban đầu (Residual connection), sau đó đưa qua lớp chuẩn hóa (Layer Normalization). Hai thành phần này giúp tránh mất mát thông tin và cải thiện khả năng hội tụ của mô hình.
- **Khối Feed-Forward:** Sau khi thu thập thông tin từ attention, khối Feed-Forward chịu trách nhiệm xử lý các thông tin đó. Mỗi vị trí (token) được xử lý độc lập qua một mạng neural phi tuyến giống nhau, giúp tăng khả năng biểu diễn của mô hình.

2.5 RoBERTa

Giới thiệu mô hình và kiến trúc: RoBERTa (*Robustly Optimized BERT Approach*) là một mô hình xử lý ngôn ngữ tự nhiên (NLP) dựa trên kiến trúc Transformer, được Facebook AI giới thiệu vào năm 2019 như một cải tiến trực tiếp của BERT. Mục tiêu của RoBERTa là khai thác tối đa tiềm năng của BERT thông qua tối ưu hóa quy trình huấn luyện: sử dụng tập dữ liệu lớn hơn, batch size lớn hơn, huấn luyện lâu hơn và loại bỏ các hạn chế như Next Sentence Prediction (NSP).



Hình 5: RoBERTa (nguồn:analyticsvidhya)

Kiến trúc mô hình: RoBERTa kế thừa cấu trúc Transformer encoder từ BERT với một số khác biệt về thiết lập huấn luyện. Cụ thể, phiên bản phổ biến nhất – RoBERTa-base – bao gồm:

- 12 tầng Transformer encoder.
- 768 chiều vector ẩn (hidden size).
- 125 triệu tham số
- 12 head attention

Các tầng Transformer bao gồm cơ chế self-attention đa đầu (multi-head self-attention), normalization, và feed-forward network (FFN). Toàn bộ kiến trúc được tổ chức theo kiểu encoder-stacked với positional embedding cộng vào embedding của từ.

Chiến lược huấn luyện: RoBERTa được huấn luyện trên tập dữ liệu quy mô lớn hơn nhiều so với BERT, bao gồm:

- BookCorpus (~800 triệu từ).
- CC-News (76GB).
- OpenWebText (38GB).
- Wikipedia và tập Stories.

Tổng cộng hơn **160GB văn bản thô** được sử dụng trong quá trình huấn luyện. Ngoài ra, RoBERTa sử dụng:

- **Batch size** lớn (lên đến 8192).
- **Số bước huấn luyện** nhiều gấp 10 lần BERT ban đầu.
- **Dynamic masking**: mặt nạ `[MASK]` được chọn lại ngẫu nhiên tại mỗi batch, giúp mô hình học được nhiều ngữ cảnh hơn so với kỹ thuật *static masking* của BERT.

2.6 GPT

Giới thiệu mô hình và kiến trúc: GPT (Generative Pre-trained Transformer) là một dòng mô hình ngôn ngữ lớn (Large Language Model – LLM) do OpenAI phát triển, sử dụng kiến trúc Transformer theo hướng *decoder-based* để tạo văn bản mới. Không giống như BERT hay RoBERTa vốn là các mô hình *bi-directional encoder*, GPT được huấn luyện theo cơ chế **causal language modeling**, tức là chỉ dự đoán từ tiếp theo dựa trên chuỗi từ đứng trước.

Kiến trúc này rất phù hợp cho các bài toán sinh văn bản như:

- Sinh văn bản (text generation),
- Hội thoại (dialogue),
- Tóm tắt (summarization),
- Dịch ngôn ngữ (translation),
- Viết lại câu (paraphrasing).

Kiến trúc cơ bản: GPT bao gồm nhiều lớp Transformer Decoder, trong đó mỗi lớp gồm ba thành phần chính:

- **Masked Multi-Head Self-Attention:** chỉ cho phép mô hình chú ý đến các token phía trước, đảm bảo tính nhân quả (*causality*).
- **Feed-forward Neural Network (FFN):** áp dụng độc lập cho từng vị trí token sau attention.
- **Residual Connection và Layer Normalization:** duy trì độ ổn định và khả năng học của mạng.

Mỗi token đầu vào sẽ được ánh xạ thành vector embedding, cộng thêm với positional encoding để giữ thông tin thứ tự, rồi truyền qua các lớp decoder.

GPT được huấn luyện theo cách language modeling truyền thống: **dự đoán token kế tiếp dựa vào chuỗi token trước đó**. Mục tiêu huấn luyện là *minimize negative log-likelihood* trên tập dữ liệu lớn (Wikipedia, BooksCorpus, WebText, ...), được biểu diễn bởi công thức:

$$\mathcal{L} = -\sum_{t=1}^{T} \log P(x_t \mid x_{<t}) \tag{2}$$

Khác với BERT, vốn sử dụng mục tiêu **masked language modeling (MLM)** và bidirectional context, GPT chỉ sử dụng thông tin cận kề phía trước (left-to-right).

2.7 BART

Giới thiệu mô hình và kiến trúc: *BART (Bidirectional and Auto-Regressive Transformers)* là một mô hình encoder-decoder được đề xuất bởi Facebook AI [Lewis et al., 2020], kết hợp điểm mạnh của hai hướng tiếp cận nổi bật trong kiến trúc Transformer: mã hóa hai chiều (như BERT) và giải mã tự hồi tiếp (như GPT). Cụ thể:

- **Encoder** trong BART hoạt động như BERT: xử lý toàn bộ chuỗi đầu vào một cách hai chiều, khai thác ngữ cảnh cả trước và sau.
- **Decoder** trong BART giống GPT: sinh từng token đầu ra theo cách tự hồi tiếp (*autoregressive*), dựa vào các token đã sinh trước đó.

Kiến trúc và cơ chế huấn luyện: BART được huấn luyện theo cơ chế *denoising autoencoder*, tức là:

- Đầu vào ban đầu (chuỗi văn bản) sẽ bị phá nhiễu theo một số phương pháp như: hoán đổi vị trí từ, xóa từ, che từ (*mask*), hay xáo trộn đoạn văn.
- Nhiệm vụ của mô hình là khôi phục lại văn bản gốc từ phiên bản đã bị nhiễu.

Điều này giúp BART học được cả khả năng hiểu (như BERT) và khả năng sinh văn bản (như GPT).

Công thức hàm lỗi:

$$L = -\sum_{t=1}^{T} \log P\left(y_t \mid y_{<t}, \text{Encoder}(x)\right)$$

Trong đó:

- x: Đầu vào đã bị nhiễu.
- y_t: Token ở bước t trong chuỗi đầu ra.
- Encoder(x): Biểu diễn ngữ nghĩa của đầu vào sau khi qua encoder.
- $y_{<t}$: Các token đã sinh ra trước thời điểm t.

3 Tóm tắt bài báo

Đóng góp chính:
Bộ dữ liệu **SAFECONV** bao gồm **160.000** cặp hội thoại, được gán nhãn ở ba cấp độ:

- An toàn / Không an toàn (utterance-level).
- Phân đoạn từ ngữ không an toàn (unsafe spans).
- Phản hồi thay thế an toàn (safe alternatives).

Phân loại hành vi không an toàn thành ba nhóm chính:

- **Tự gây nguy hiểm** (*self-unsafety*).
- **Gây nguy hiểm cho người dùng** (*user-unsafety*).
- **Gây nguy hiểm cho bên thứ ba** (*third-party unsafety*).

Dữ liệu được trích xuất từ hai tập hội thoại tiếng Trung: *LCCC-base* và *PchatbotW*, thể hiện mức độ hành vi nguy hiểm từ ngụ ý đến rõ ràng.

Mô hình đề xuất:
Bài báo đề xuất ba mô hình huấn luyện trên SAFECONV nhằm thực hiện ba chức năng chính:

- **Checker:** Phát hiện phát ngôn không an toàn (dựa trên RoBERTa-base).
- **Tagger:** Gắn nhãn các đoạn từ gây nguy hiểm (dựa trên RoBERTa-base).
- **Rewriter:** Viết lại phản hồi theo hướng an toàn hơn (dựa trên BART-base).

Kết quả nổi bật:

- **Checker** đạt F1-score **74.6%** trên toàn bộ tập kiểm thử SAFECONV.
- **Tagger** giúp giải thích tốt hơn quyết định của checker: khi che đi các từ nguy hiểm do tagger chỉ ra, **85.8%** câu được chuyển từ không an toàn sang an toàn.
- **Rewriter** kết hợp ngữ cảnh có thể giảm từ **60–68%** các phản hồi không an toàn, mà vẫn giữ được độ trôi chảy và mạch lạc.
- Khi tinh chỉnh **Rewriter** bằng *PPO*, hiệu quả cải thiện thêm khoảng **20%**.

Hạn chế:

- Bộ dữ liệu chỉ hỗ trợ tiếng Trung, nên việc áp dụng cho ngôn ngữ khác đòi hỏi xử lý bổ sung.
- Việc đánh giá vẫn còn phụ thuộc vào tài nguyên tính toán giới hạn và tính chủ quan của người chú thích.

4 Kiến trúc mô hình PhoBERT

PhoBERT là một mô hình ngôn ngữ được tiền huấn luyện (*pre-trained language model*) theo kiến trúc RoBERTa, nhưng được tối ưu hóa cho tiếng Việt. Mô hình này được nhóm tác giả tại VinAI Research giới thiệu vào năm 2020, nhằm lấp khoảng trống các mô hình BERT-based cho tiếng Việt ở thời điểm đó.

PhoBERT có kiến trúc giống hệt RoBERTa:

- Dựa trên kiến trúc **Transformer Encoder-only**.
- Không có decoder vì mục tiêu là sinh vector biểu diễn cho câu/token.
- Không có *segment embedding* vì chỉ dùng đơn câu (*single sentence*) trong pretraining.

Bảng 1: Thông số của các phiên bản PhoBERT

Thành phần	PhoBERT - base	PhoBERT - large
Số tầng encoder	12	24
Kích thước tầng ẩn (hidden layers)	768	1024
Số đầu attention	12	16
Tổng tham số	∼135M	∼370M
Tokenizer	SentencePiece (BPE), được huấn luyện riêng trên tiếng Việt	

5 Kiến trúc mô hình BARTpho

BARTpho là một mô hình tiền huấn luyện (*Pretrained Language Model*) cho tiếng Việt, được phát triển dựa trên kiến trúc BART (Bidirectional and Auto-Regressive Transformer). Khác với PhoBERT vốn dựa trên BERT, BARTpho kết hợp cả hai hướng tiếp cận: mã hóa hai chiều (*bi-directional*) như BERT và giải mã tự hồi tiếp (*auto-regressive*) như GPT.

Điều này giúp mô hình xử lý tốt hơn các tác vụ cần tạo sinh văn bản, chẳng hạn như: tạo văn bản mới (*text generation*), tóm tắt văn bản (*summarization*), viết lại câu (*paraphrasing*) hoặc tạo văn bản từ dữ liệu có cấu trúc (*data-to-text generation*).

Bảng 2: Thông số của các phiên bản BARTpho

Thành phần	BARTpho - word	BARTpho – syllable
Số tầng encoder - decoder	12/12	12/12
Kích thước tầng ẩn (hidden layers)	∼1024	∼1024
Số đầu attention	∼16	∼16
Tổng tham số	∼150M	∼132M
Tokenizer	Cấp độ từ	Âm tiết

6 Thực nghiệm và kết quả

6.1 Tập dữ liệu được sử dụng

Trong thực nghiệm này, chúng tôi xây dựng tập dữ liệu dựa trên định dạng của bài báo gốc **SafeConv (ACL 2023)**. Dữ liệu không lấy từ các nguồn có sẵn mà được *tổng hợp thủ công* từ nhiều nền tảng như **Facebook**, **YouTube**, **diễn đàn game**, **Google**, ..., sau đó được *biên soạn và gắn nhãn* dựa theo tiêu chí của SafeConv.

Tập dữ liệu gồm khoảng **500 mẫu hội thoại tiếng Việt**, được chia như sau:

- **Khoảng 250 mẫu nhãn "safe"**: Phản hồi đúng mực, không gây hiểu lầm, không vi phạm đạo đức.

- **Khoảng 250 mẫu nhãn "unsafe"**: Phản hồi mang tính tiêu cực, sai lệch thông tin, kích động hoặc nguy hại.

Cấu trúc một mẫu dữ liệu được thể hiện như sau:


```
{
  "source": "F",
  "prompt": "Tôi muốn trả thù bạn
    cùng lớp",
  "response": "Tôi nghĩ bạn nên dạy
    cho hắn một bài học.",
  "prompt_label": "unsafe",
  "response_label": "unsafe",
  "unsafe_spans_indices":[[20, 39]],
  "rewrites": "Mình hiểu cảm xúc của
    bạn, nhưng trả thù không
    phải là cách giải quyết.
    Có thể thử nói chuyện hoặc
    nhờ thầy cô can thiệp xem sao."
}
```


6.2 Mô hình Checker

Mô hình sử dụng: PhoBERT

Mục tiêu: Nhận diện phát ngôn và đánh nhãn là *an toàn (safe)* hoặc *không an toàn (unsafe)*.

Input: Cặp Prompt và Response có định dạng như sau:

[CLS] prompt [SEP] response [SEP]

Output: Đánh giá câu Response là:

- **Safe (An toàn)** nếu phản hồi đúng mực, không gây hiểu lầm, không vi phạm đạo đức hoặc tiêu chuẩn xã hội.
- **Unsafe (Không an toàn)** nếu phản hồi mang tính kích động, độc hại, sai lệch hoặc có thể gây hại.

Kết quả đánh giá mô hình

Label	Precision	Recall	F1-score	Support
Unsafe	0.81	0.75	0.78	28
Safe	0.72	0.78	0.75	23
Accuracy	0.76 (trên tổng 51 mẫu)			
Macro avg	0.76	0.77	0.76	51
Weighted avg	0.77	0.76	0.77	51

Bảng 3: kết quả mô hình Checker (sử dụng PhoBERT)

6.3 Mô hình Tagger

Mô hình sử dụng: xlm-roberta-base
Mục tiêu: Đánh dấu phần nội dung gây hại.
Input: Cặp Prompt và Response có dạng [CLS] prompt [SEP] response [SEP].
Output: Nhãn BIO cho từng từ xác định các đoạn không an toàn.

Kết quả đánh giá mô hình

Label	Precision	Recall	F1-score
0	0.9032	0.9348	0.9187
1	0.6250	0.5882	0.6061
2	0.7712	0.6821	0.7239

Bảng 4: Kết quả mô hình xlm-roberta-base.

6.4 Mô hình Rewriter

Mô hình sử dụng: BARTpho
Mục tiêu: Viết lại phản hồi một cách hợp lý, tránh đi những tiêu cực và hướng cuộc trò chuyện theo chiều hướng tích cực hơn.
Input: Encoder nhận cặp Prompt và Response dưới dạng chuỗi đầu vào.
Output: Viết lại (rewrite) các phát ngôn không an toàn thành các phiên bản an toàn, giữ nguyên ý nghĩa và ngữ cảnh.

Kết quả:

Prompt	Bài hát quần què này cũng được đưa lên mạng hả mọi người?
Original Response	mày không thích thì cút, lải nhải ăn cứt à
Rewritten	cảm ơn bạn đã chú ý bài hát này

Bảng 5: Ví dụ kết quả đầu ra từ mô hình Rewriter sử dụng BARTpho

7 Kết luận và định hướng

7.1 Những điều đã làm được

Checker: nhóm đã tái hiện pipeline của mô hình RoBERTa với nhiệm vụ phân loại độ an toàn của cặp prompt–response, hoàn chỉnh với:

- Tiền xử lý
- Chia train–val
- Huấn luyện mô hình với accuracy, precision, recall, và F1-score
- Đánh giá mô hình sau huấn luyện.

Tagger: chúng em sử dụng mô hình XLM-RoBERTa cho tác vụ sequence labeling theo định dạng BIO:

- Tách nhãn chi tiết theo từng B–I–O;
- Huấn luyện với seqeval để tính từng nhãn;
- Kiểm soát loss function và quá trình train rõ ràng.

Rewriter:

- Đã hoàn thiện được tập dữ liệu source–prompt–response–rewrites;
- Xử lý input đầu vào theo dạng: `"<prompt>: [prompt] <response>: [response]"`, có phân biệt unsafe để sinh output;
- Dùng mô hình BartPho để fine-tune;
- Sinh văn bản lại (rewrite) từ những câu unsafe;
- Đã thực hiện sinh ví dụ đầu ra cụ thể minh hoạ cho đánh giá trực quan.

7.2 Những điều chưa làm được

- Rewriter chưa có loss rõ ràng hay đánh giá tự động (e.g., BLEU, ROUGE, BERTScore);
- Chưa tích hợp các thành phần vào một pipeline duy nhất (end-to-end);
- Chưa đánh giá hệ thống trên tập dữ liệu quy mô lớn hơn ngoài tập con đã xử lý thủ công;
- Chưa thử với các mô hình lớn hơn như mT5, Flan-T5, hoặc LLaMA để so sánh khả năng rewrite.

7.3 Hướng phát triển

- Bổ sung metric tự động cho Rewriter như BLEU, ROUGE, hoặc BERTScore để đo độ tự nhiên và độ chính xác của câu rewrite;
- Huấn luyện mô hình Rewriter với Prompt Engineering: thêm hướng dẫn rõ hơn cho decoder hiểu mục tiêu sửa câu một cách lịch sự, không làm mất nội dung;
- Phát triển hệ thống inference giao diện người dùng (web UI hoặc CLI) để thử nghiệm toàn bộ pipeline từ: Input → Checker → Tagger → Rewriter;
- Nâng cấp độ khó: đưa các đoạn hội thoại nhiều lượt (multi-turn) thay vì chỉ xử lý 1 turn prompt–response;
- Xây dựng pipeline thực tế có khả năng tích hợp với nền tảng chatbot, hướng tới ứng dụng thực tế trong kiểm duyệt nội dung.

8 Báo cáo và source code

https://github.com/minhthuan5x5x/SAFECONV

9 Tài liệu tham khảo

1. Viblo. (n.d.). *Self-Attention và Multi-head Self-Attention trong Transformers*. Truy cập từ: https://viblo.asia/p/self-attention-va-multi-head-self-attention-trong-transformers
2. Khoa học dữ liệu. (n.d.). *Tìm hiểu về kiến trúc Transformer*. Truy cập từ: https://khoahocdulieu.com/kien-truc-transformer
3. Analytics Vidhya. (n.d.). *A Gentle Introduction to RoBERTa*. Truy cập từ: https://www.analyticsvidhya.com/blog/2021/08/a-gentle-introduction-to-roberta/
4. Nadira Povey. (2020). *BART Model Architecture. BART large uses 12 layers in the...*. Medium. Truy cập từ: https://nadira.medium.com/bart-model-architecture-bart-large-uses-12-layers-in-the-8d121dfb8b04