SỬ DỤNG BERT ĐỂ TÓM TẮT VĂN BẢN

Mục lục

1 Giới thiệu bài toán

Trong thế giới giàu thông tin ngày nay, việc đọc và hiểu các tài liệu dài thường rất khó khăn. Do đó, tóm tắt văn bản có thể là một công cụ giúp hiểu nhanh các ý chính của một bài báo, một văn bản hoặc bất kỳ đoạn tài liệu nào. Văn bản sau khi tóm tắt phải đảm bảo giữ được những thông tin quan trọng của văn bản gốc cũng như tính đúng đắn về chính tả và ngữ pháp.
Theo phương pháp thực hiện, tóm tắt văn bản được chia làm hai loại:

- Tóm tắt kiểu trích chọn - extraction: chọn ra những từ, cụm từ hoặc câu đã có sẵn và quan trọng nhất của văn bản, sau đó kết hợp chúng lại để tạo ra văn bản tóm tắt.

- Tóm tắt kiểu tóm lược ý - abstraction: sử dụng các từ, cụm từ hoặc câu mới, có thể khác với văn bản gốc nhưng vẫn giữ được ý nghĩa tương đồng để xây dựng văn bản tóm tắt.

Tóm tắt tóm lược tạo ra một bản tóm tắt hiệu quả hơn so với tóm tắt trích chọn bởi việc nó có thể trích chọn những thông tin từ các văn bản gốc để khởi tạo bản tóm tắt thông tin rõ ràng. Tuy nhiên phương pháp này khá phức tạp nên trong bài báo cáo này, em sử dụng hướng tiếp cận trích chọn extrction để tạo bản tóm tắt.

2 Cơ sở lý thuyết

Hiện nay có khá nhiều thuật toán và mô hình đã được triển khai để xử lý ngôn ngữ tự nhiên cũng như tóm tắt văn bản: RNN, Transformer, ... Và trong đồ án này, em sẽ sử dụng BERT - một mô hình biểu diễn ngôn ngữ để xử lý bài toán tóm tắt văn bản.

BERT là một cải tiến của kiến trúc Transformer. Transformer sử dụng cả bộ mã hóa encoder và bộ giải mã decoder đễ nhận và xử lý thông tin thì BERT chỉ sử dụng bộ mã hóa. Và BERT chính là mô hình điển hình sử dụng cơ chế chú ý

- thành phần chính tạo nên sự đột phá của Transformer trong các bài toán xử lý của NLP so với RNN.

Sau đây em xin trình bày một số khái niệm và mô hình cơ bản.

2.1 Cơ chế attention

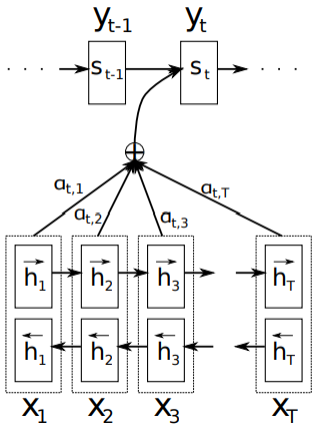
Cơ chế attention (attention mechanism) [1] trong học sâu đã mang lại hiệu quả đáng kể cho nhiều mô hình, nó đã và đang tiếp tục là một thành phần không thể thiếu trong các mô hình tiến tiến nhất. Cơ chế attention là một kỹ thuật cho phép mô hình tập trung vào các phần khác nhau của đầu vào tại mỗi bước thời gian.

Trong bài toán dịch máy (Neural Machine Translation - NMT), ta thường sử dụng mô hình seq2seq [**?**]. Khối Encoder nén thông tin đầu vào thành một vector. Vector biểu diễn này sẽ mang toàn bộ thông tin của đầu vào, được đưa vào decoder để tạo ra câu đích. Tuy nhiên, khi độ dài chuỗi tăng thì chất lượng của mô hình sẽ giảm đáng kể. Đối với chuỗi dài, việc phải nén toàn bộ thông tin của chuỗi đầu vào thành một vector duy nhất có thể dẫn đến việc bị mất thông tin. Ngoài ra, decoder chỉ nhận một vector biểu diễn đầu vào duy nhất và sẽ phải trích các thông tin liên quan này từ một vector biểu diễn duy nhất này - việc này cũng vô cùng khó.

Để giải quyết vấn đề trên, cơ chế attention đã được ra đời vào năm 2015.

2.1.1 Ý tưởng

Cơ chế attention cho phép mô hình tập trung vào các phần khác nhau của câu nguồn khi dịch từng từ hoặc cụm từ trong câu đích bằng cách sử dụng vector ngữ cảnh thay vì sử dụng vector trạng thái ẩn cuối cùng của encoder để tạo vector biểu diễn cho decoder. Vector ngữ cảnh có thể tương tác với tất cả các vector trạng thái ẩn của encoder.



Hình 1: *Kiến trúc của cơ chế attention*

2.1.2 Cách hoạt động

- Nhận vector trạng thái ẩn của quá trình lan truyền ngược (backward hidden state) $\overleftarrow{h_t}$ của quá trình lan truyền xuôi (forward hidden state) $\vec{h_t}$ và vector trạng thái ẩn của decoder s_t.
- Tính toán điểm attention: Đối với mỗi từ đích đang được dịch, mô hình sẽ tính toán một điểm số (score) giữa trạng thái ẩn hiện tại của từ đích và tất cả các trạng thái ẩn của câu nguồn.

$$e_{ij} = a(s_{t-1}, h_j),$$

trong đó $h_j = [\vec{h_t}, \overleftarrow{h_t}]^T$. Có một số cách để tính điểm attention:

  - Dot-product attention:

$$\mathrm{a}(h_t, s_i) = h_t^T s_i$$

  - Scaled Dot-product Attention:

$$\mathrm{a}(h_t, s_i) = \frac{h_t^T s_i}{\sqrt{d_k}}$$

    trong đó d_k là kích thước của key vectors

– Additive Attention:

$$\mathrm{a}(h_t, s_i) = v^T \tanh\left(W_{h_t} h_t + W_{s_t} s_t\right)$$

- Tính trọng số attention: để xác định mức độ "chú ý" mà từ đích cần dành cho mỗi từ nguồn

$$\alpha_{ij} = \mathrm{softmax}(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

- Tính vector context: bằng tổng trọng số của các vector trạng thái ẩn của câu nguồn, với trọng số được xác định bởi phân phối attention

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

- Dự đoán từ tiếp theo: Vector context này sau đó được kết hợp với trạng thái ẩn hiện tại của từ đích để dự đoán từ tiếp theo trong câu đích.

2.1.3 Hạn chế

Với những chuỗi dài, số lượng cặp phần tử tăng theo cấp số nhân, làm tăng đáng kể độ phức tạp tính toán. Cụ thể, nếu có N phần tử trong chuỗi, ta cần tính toán N^2 trọng số attention. Điều này dẫn đến một tình trạng "nút cổ chai" vì quá trình tính toán trọng số attention trở nên rất tốn kém về mặt tính toán, đặc biệt là khi N là một giá trị lớn.

2.1.4 Self-attention

Để giải quyết những hạn chế của attention, self-attention đã được ra đời. Self-attention cho phép mỗi từ trong chuỗi "chú ý" đến tất cả các từ khác trong chuỗi, kể cả chính từ đó, với các trọng số attention được học. Điều này giúp mô hình tự nâng cao khả năng tái sử dụng thông tin và linh hoạt hơn trong việc xử lý chuỗi

dài. Mỗi từ có thể chú ý đến các từ quan trọng khác nhau trong mỗi bước, không tạo ra một "nút cổ chai" duy nhất.

Self-attention được áp dụng để học biểu diễn của một từ với tất cả các từ trong chuỗi, kể cả chính nó. Cơ chế:

- Tạo vector Query Q, Key K và Value V.
  Query được sử dụng khi một token "quan sát" những tokens còn lại, nó sẽ tìm kiếm thông tin xung quanh để hiểu được ngữ cảnh và mối quan hệ của nó với các tokens còn lại. Key sẽ phản hồi yêu cầu của Query và được sử dụng để tính attention weight. Cuối cùng, Value sử dụng attention weight vừa rồi để tính ra vector đại diện (attention vector).

  $$Q = X * W^Q, K = X * W^K, V = X * W^V$$

  trong đó W^Q, W^K, W^V được khởi tạo ngẫu nhiên và được cập nhật thông qua quá trình lan truyền ngược, X là vector đặc trưng của đầu vào.

- Tính toán attention scores.
  Sử dụng hàm scaled dot-product attention

  $$a(Q, K) = \frac{QK^T}{\sqrt{d_k}}$$

  với d_k là số chiều của vector Key. Phân số chia $\sqrt{d_k}$ để khắc phục hiện tượng vanishing gradient khi tính toán gradient trong quá trình lan truyền ngược.

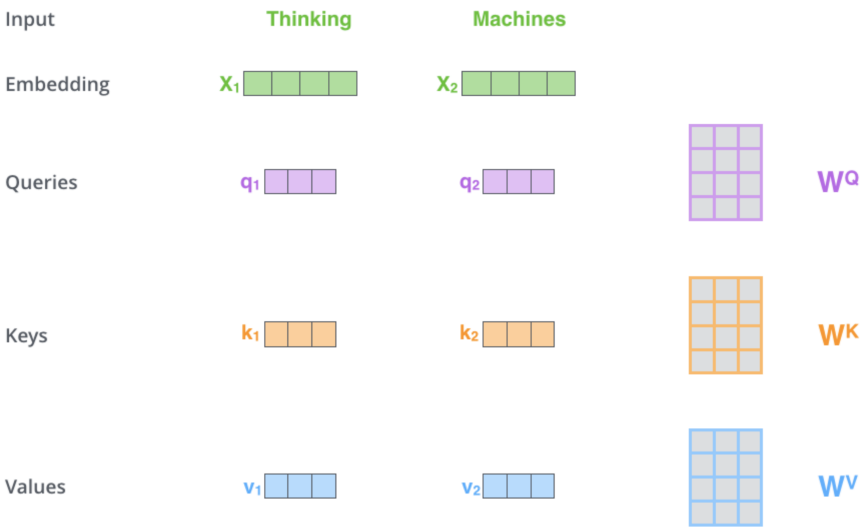
- Áp dụng softmax

  $$\text{softmax}(a(Q, K)) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})$$

  để chuẩn hóa attention scores

- Tính toán đầu ra

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

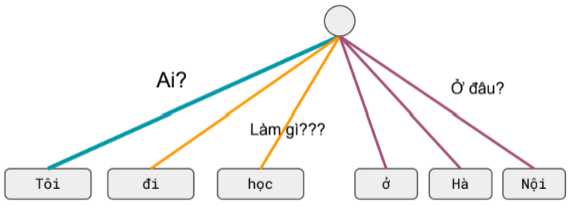


Hình 2: Tổng hợp quá trình self-attention

2.1.5 Multi-head attention

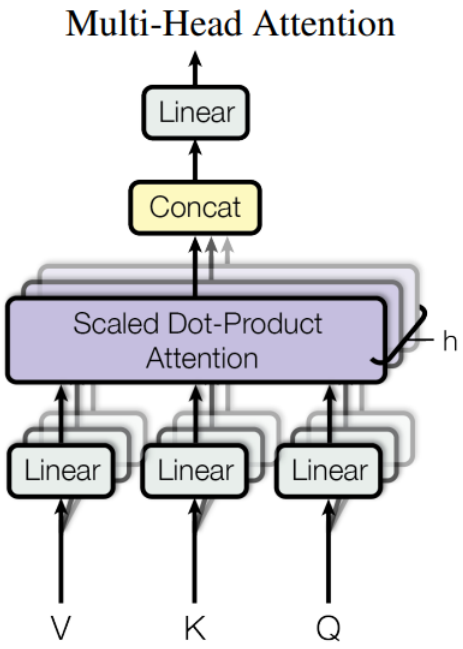
Khi sử dụng duy nhất một self-attention, mô hình thực hiện biến đổi tuyến tính trên toàn bộ vector biểu diễn từ nên không có khả năng bóc tách từng phần nhỏ thông tin tại các khu vực nhất định.

Và để giải quyết vấn đề đó, mô hình sẽ sử dụng cơ chế multi-head attention [2] gồm nhiều "head" self-attention song song, mỗi "head" này sẽ phụ trách học một phần thông tin của câu.



Ví dụ: attetion head 1 học embedding của từ "tôi" để học cách đặt câu hỏi "Ai?". Attention head 2 học embedding của từ "đi" và từ "học" để học cách đặt

câu hỏi "Làm gì?". Attention head 3 học embedding của từ "ở" và từ "Hà Nội" để học cách đặt câu hỏi "Ở đâu?".



Hình 3: Multi-head attention

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

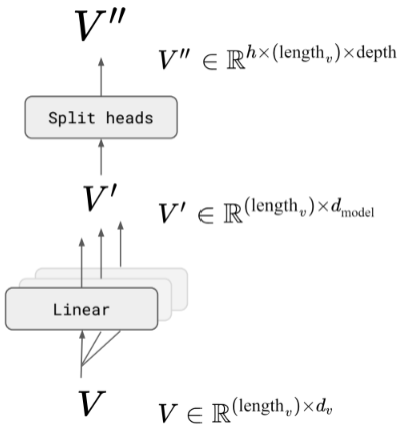
trong đó

- $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$
- $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$
- $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$
- $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$

- $W^O \in \mathbb{R}^{d_{\text{model}} \times hd_v}$

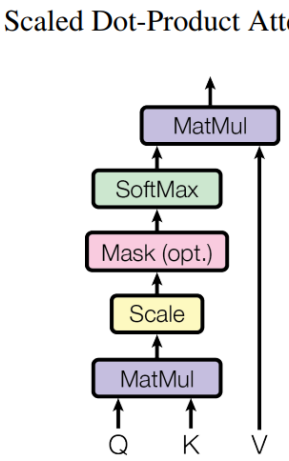
Multi-head attention gồm 4 lớp:

- Lớp tuyến tính đầu tiên: chuyển Query, Key và Value và cùng một chiều (để có thể thực hiện phép nhân khi tính head_i). Làm tương tự với Q và K.



Hình 4: Linear layers and split into heads

- Scale dot-product attetion: tính trọng số attention



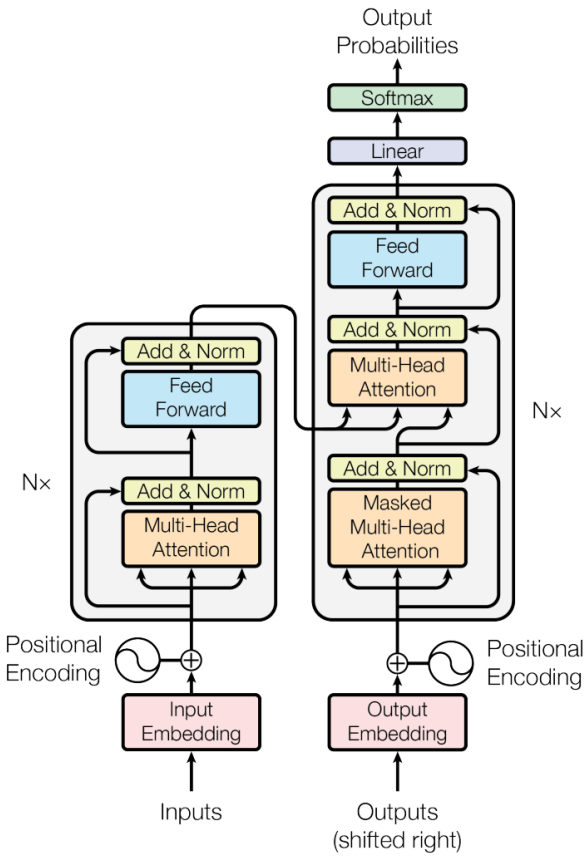
Hình 5: Scale dot-product attention cho từng lớp

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

Việc chia cho $\sqrt{d_k}$ để đảm bảo rằng giá trị tích attention không quá lớn khi số chiều d lớn.

- Lớp concat và linear cuối cùng: mô hình nối kết quả attention trên từng head với nhau và đưa về cùng chiều với chiều của đầu vào.

2.2 Kiến trúc Transformer



Hình 6: Kiến trúc Transformer

Gồm 2 phần chính: Encoder và Decoder **Do BERT chỉ sử dụng encoder nên trong báo cáo này, em không trình bày về cấu trúc của decoder.**

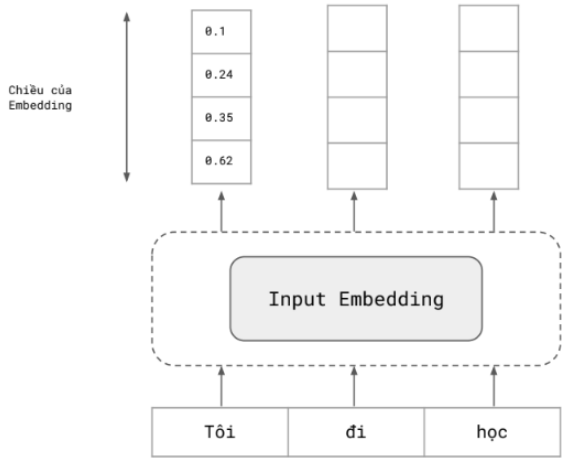
2.2.1 Encoder

- Nhiệm vụ của Encoder: Encoder học mối tương quan giữa mỗi từ trong câu với các từ còn lại và chính nó, sau đó sẽ bổ sung mối tương quan này vào Embedding của từng từ đầu.

- Encoder gồm 2 phần chính
  - Multi-head attention
  - Positionwise fully connected feed-forward network

Sau đây em trình bày kiến trúc của một lớp trong encoder.

a. Input Embedding:
Chuẩn hóa mỗi từ trong câu thành một vector.



Hình 7: Input embedding

b. Positional Encoding:
Transformer không có các RNN hay CNN nên mô hình không thể đảm bảo thứ tự của các từ trong chuỗi đầu vào (do trong Transformer, các từ sẽ được đưa vào cùng một lúc nên không biết từ nào trước từ nào sau). Do đó, positional encodings bổ sung thông tin vị trí của từng từ qua công thức:

$$PE_{pos,2i} = \sin(pos/10000^{2i/d_{model}}),$$

$$PE_{pos,2i+1} = \cos(pos/10000^{2i/d_{model}})$$

với d_{model} là kích thước của Embedding, *pos* là vị trí của token, i là vị trí theo chiều Embeeding.
Sau đó, vector positional encoding được cộng với input embedding trước khi đưa vào mô hình.

c. Lớp Add & Norm:
Transformer sử dụng Residual-skip connection để thêm input vào output của lớp multi-head attention và sử dụng Layer norm để chuẩn hóa output sau khi đã thêm thông tin của input.

$$\text{LayerNorm}(Z) = \gamma \frac{Z - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta$$

trong đó

- Z là vector đầu vào với số chiều d
- $\mu = E[Z]$, $\sigma^2 = Var[Z]$
- γ là tham số scale
- β là tham số để điều chỉnh trung bình của output.
- ϵ là 1 hằng số nhỏ được thêm vào mẫu để tránh phép chia cho 0.

d. Residual - skip connecttion:
là một kỹ thuật thêm đầu vào của khối vào đầu ra của nó. Kết nối này giúp mang thông tin vị trí từ layer thấp đến layer cao do đó thông tin vị trí sẽ được giữ đến output, nhằm hạn chế hiện tượng biến mất của gradient .

e. Position-wise Feed-Forward Networks [3]:
Sau khi self-attention đã tạo ra một biểu diễn tương quan cho mỗi từ, position-wise FFN sẽ xử lý và biến đổi biểu diễn này tại mỗi vị trí riêng lẻ trong chuỗi.
Positionwise FFN gồm 2 lớp full-connected và 1 hàm kích hoạt ở giữa.

$$\text{FFN}(x) = \text{ReLU}(xW_1 + b_1)W_2 + b_2$$

2.3 BERT

2.3.1 Tổng quan

BERT [4] - Bidirectional Encoder Representations from Transformers, là một mô hình ngôn ngữ dựa trên kiến trúc Transformer được giới thiệu bởi Google vào năm 2018. BERT đã mang lại một bước tiến đột phá trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP) nhờ khả năng hiểu ngữ cảnh hai chiều của nó.

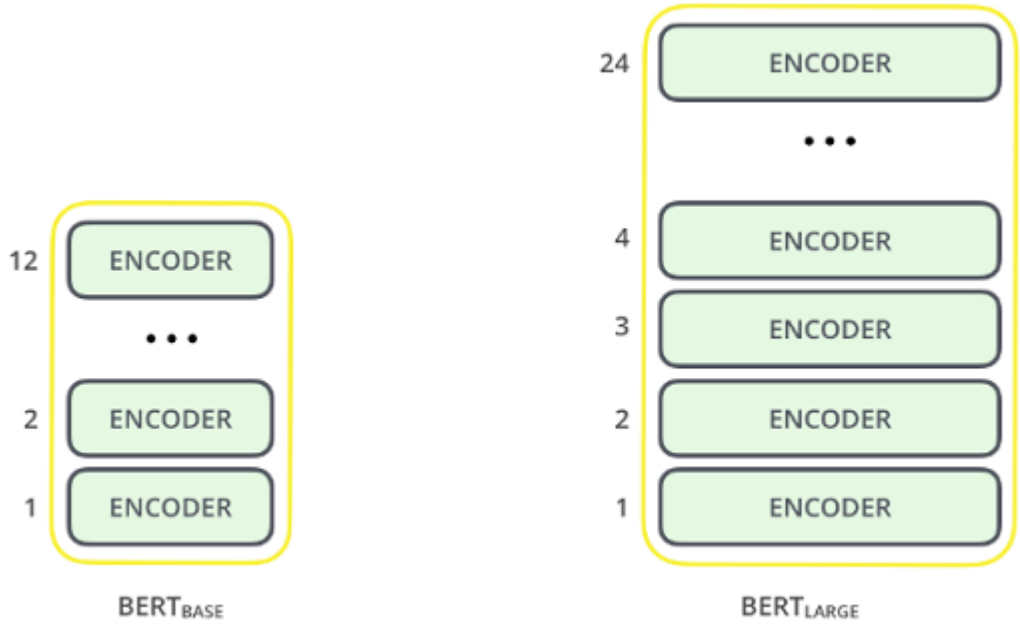
2.3.2 Ưu điểm

Các mô hình ngôn ngữ trước đây thường chỉ mã hóa ngữ cảnh theo một hướng duy nhất - từ trái sang phải hoặc từ phải sang trái. Tuy nhiên, BERT đã thay đổi điều này bằng cách sử dụng kiến trúc Transformer để mã hóa ngữ cảnh theo cả hai chiều. Điều này giúp BERT có cái nhìn sâu sắc hơn về ngữ cảnh của từ trong câu, cho phép nó hiểu rõ hơn nghĩa của từ dựa trên ngữ cảnh xung quanh.

Ngoài ra, BERT là một trong những "đại diện" ưu tú trong việc sử dụng kỹ thuật transfer learning cho NLP. Transfer learning là một kỹ thuật mà mô hình được đào tạo trước (pre-trained) trên một tập dữ liệu lớn và được tinh chỉnh (fine-tune) cho các bài toán hoặc tập dữ liệu cụ thể. Kỹ thuật này giúp giảm đáng kể chi phí đào tạo mô hình và thời gian.

2.3.3 Kiến trúc

BERT kế thừa kiến trúc của Transformer nhưng chỉ sử dụng bộ mã hóa Encoder.



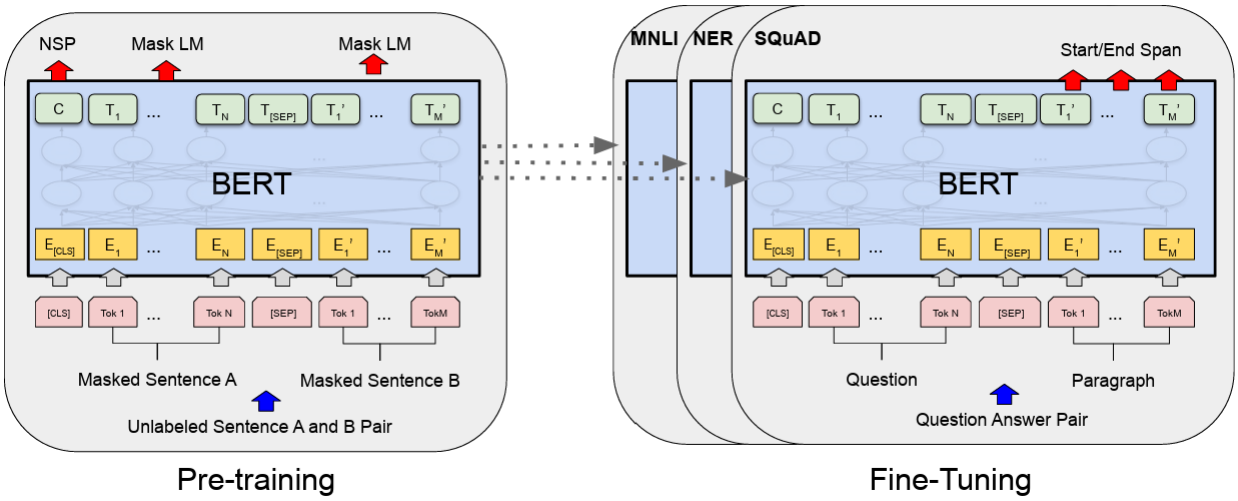
Hình 8: Kiến trúc của BERT gồm các khối Encoder xếp chồng lên nhau

- $\text{BERT}_{\text{BASE}}$ có $L = 12, H = 768, A = 12$, tổng tham số là 110 triệu.
- $\text{BERT}_{\text{LARGE}}$ có $L = 24, H = 1024, A = 16$, tổng tham số là 340 triệu.

trong đó, L là số lượng các block sub-layers trong transformer, H là kích thước của vector embedding (hidden size), A là số lượng head trong multi-head layer.

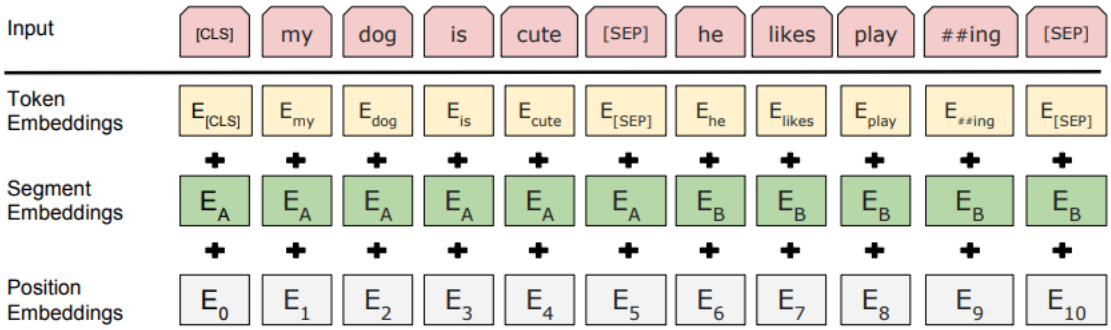
BERT gồm 2 tiến trình:

- Pre-training
- Fine-tuning



Hình 9: Tổng quan về tiến trình pre-trained và fine-tune của BERT

a. Trước khi đi vào chi tiết 2 tiến trình, ta hãy tìm hiểu cách biểu diễn input của BERT.



Hình 10: Biểu diễn input của BERT

Input có thể là biểu diễn của một câu đơn, hoặc một cặp câu được đặt thành một chuỗi. Khi có 1 chuỗi đầu vào cụ thể, biểu diễn đầu vào được tính bằng cách lấy tổng của các token (token embeddings) với vector phân đoạn

(segment embeddings) và vị trí tương ứng của các từ trong chuỗi (position embeddings).

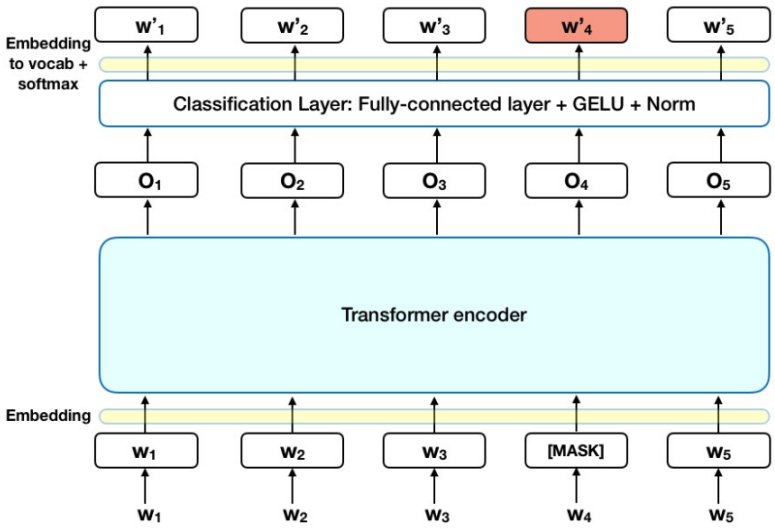
Có một số lưu ý như sau:

- Token đầu tiên của mỗi chuỗi được mặc định là [CLS]
- Sử dụng token [SEP] để phân tách các câu và thêm một segment embedding cho câu A (E_A) và một segment embedding khác cho câu B (E_B).
- Sử dụng positional embeddings với độ dài câu tối đa là 512 tokens.

Sau khi đã có biểu diễn đầu vào, ta tiếp tục với các tiến trình của BERT.

b. Pre-trained: Trong giai đoạn này, BERT được đào tạo trên một lượng lớn dữ liệu văn bản không được gán nhãn, gồm hai nhiệm vụ không giám sát:

- Masked Language Model (MLM): 1 số từ trong câu được che lại (masked) và được thay thế bằng 1 token đặc biệt, thường là "[MASK]" (thường 15% số từ trong câu được che đi). Mô hình sẽ dự đoán những từ này dựa trên ngữ cảnh còn lại của câu.



Hình 11: Nhiệm vụ MLM

Ví dụ có câu "Hôm nay trời rất đẹp". Mô hình nhận đầu vào là "Hôm nay trời rất [MASK]". Nhiệm vụ của mô hình là dự đoán từ "[MASK]" dựa trên ngữ cảnh còn lại, và từ đó học được ngữ nghĩa và ngữ cảnh của từ "đẹp".

- Next Sentence Prediction (NSP): Trong nhiệm vụ này, mô hình nhận vào hai câu và cố gắng dự đoán câu thứ hai có phải là câu tiếp theo của câu đầu tiên trong văn bản gốc hay không.

  Ví dụ có hai câu "Tôi đang đi dạo trong công viên. Thời tiết hôm nay rất đẹp." Mô hình nhận đầu vào là hai câu này và sẽ dự đoán xem câu "Thời tiết hôm nay rất đẹp" có phải là câu tiếp theo của câu "Tôi đang đi dạo trong công viên" trong văn bản gốc hay không.

MLM cho phép BERT học được ngữ cảnh hai chiều, còn NSP giúp BERT học được mối liên hệ giữa các câu với nhau. Từ đó mô hình hiểu được ngữ cảnh rộng hơn, không chỉ ở mức độ từ và câu, mà còn ở mức độ đoạn văn.

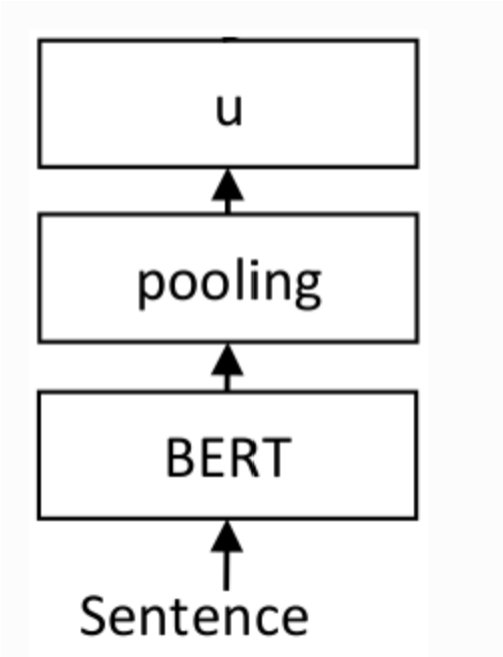
c. Fine-tune: Sau khi đã được pre-trained, BERT có thể được tinh chỉnh để thực hiện các bài toán NLP khác nhau. Trong giai đoạn này, BERT được huấn luyện trên 1 tập dữ liệu nhỏ được gán nhãn cho một nhiệm vụ cụ thể nào đó (VD: phân loại văn bản, trả lời câu hỏi, phân tích cảm xúc, ...).

3 Sentence-BERT

BERT có hạn chế là không thể embedding cả câu mà chỉ có thể embedding các token trong câu. Điều này dẫn đến khó khăn trong việc so sánh độ tương đồng giữa các câu việc so sánh sự tương đồng giữa các câu quan trọng, loại bỏ những câu có ý nghĩa tương tự nhau giúp văn bản ngắn gọn hơn và tìm kiếm những câu có ý nghĩa liên quan đến một chủ đề nào đó để tạo bản tóm tắt theo yêu cầu của người sử dụng. Để giải quyết vấn đề trên, Sentence-BERT đã ra đời.

Sentence-BERT [5], hay viết tắt là SBERT, là một biến thể của BERT sử dụng mạng siamese và mạng triplet để tạo ra các embedding câu có ý nghĩa về mặt ngữ nghĩa.

3.1 Kiến trúc của SBERT

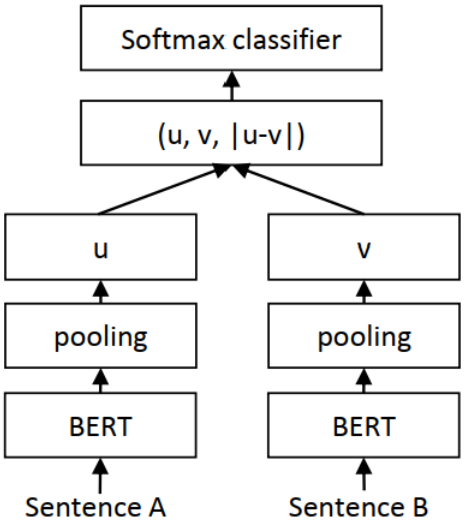


Hình 12: Kiến trúc cơ bản của SBERT

Input, có thể là một câu hoặc văn bản, được đưa vào mô hình BERT để tạo các word embeddings cho tất cả các token. Đưa các word embedding đó qua một lớp pooling để tạo vector đặc trưng có kích thước cố định, thường chọn mean pooling: tính trung bình tất cả các word embeddings để đơn giản mô hình.

Tùy vào các tác vụ cụ thể mà SBERT có thể sử dụng mạng siamese hoặc mạng triplet.

- Mạng siamese: chứa hai mạng con giống hệt nhau, có cùng tham số và trọng số. Mạng siamese được sử dụng để so sánh, đánh giá mức độ tương đồng giữa các đối tượng hoặc dùng để phân loại tương đồng.

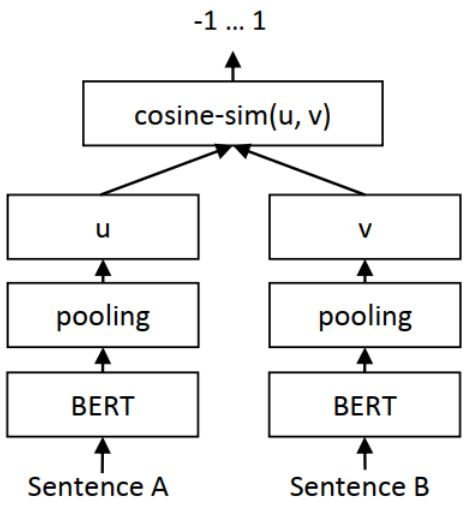


Hình 13: Kiến trúc của SBERT sử dụng mạng siamese với tác vụ phân loại mục tiêu

Trước hết, cặp câu A và B được đưa qua mô hình BERT để tạo ra vector đặc trưng, đưa các vector đặc trưng đó qua một lớp pooling (mean pooling) để tạo vector đặc trưng của câu, giúp giảm chiều dữ liệu. Sau đó, tính toán $|u - v|$ và sử dụng hàm softmax để phân loại tương đồng.

$$o = \text{softmax}(W_t(u, v, |u - v|))$$

trong đó, $W_t \in \mathbb{R}^{3n \times k}$ với n là số chiều của vector đặc trưng u/v, k là số label.



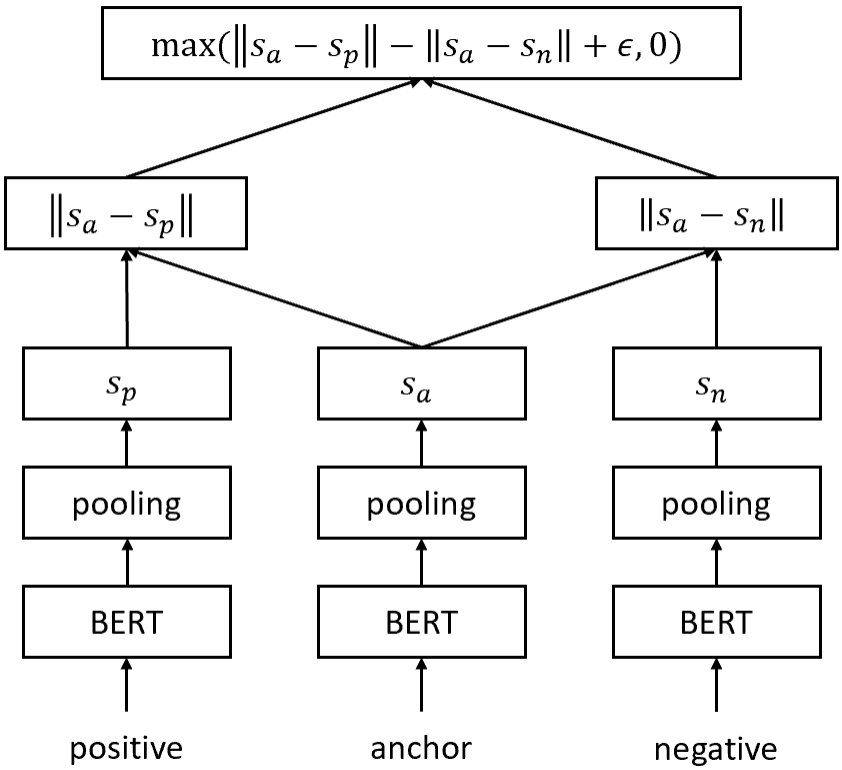
Hình 14: Kiến trúc của SBERT khi sử dụng mạng siamese để tính điểm tương đồng

Với tác vụ so sánh mức độ tương đồng, sau khi có được các vector đặc trưng u và v, ta tính toán cosine similarity

$$\text{cosine-sim} = \frac{u \cdot v}{||u||||v||}.$$

Nếu kết quả tính toán cosine-sim $\in [-1, 0)$, hai câu trái nghĩa nhau; $= 0$, hai câu không liên quan gì đến nhau; và $\in (0, 1]$, hai câu có ý nghĩa tương tự nhau, đặc biệt, nếu giá trị $= 1$ thì hai câu giống hệt nhau.

- Mạng triplet:



Hình 15: Kiến trúc của SBERT sử dụng mạng triplet

Mạng triplet sử dụng ba đầu vào cho mỗi mẫu dữ liệu: "anchor" (đại diện cho một lớp), "positive" (cùng lớp với anchor), và một mẫu "negative" (lớp khác với anchor). Mục tiêu là học biểu diễn sao cho khoảng cách giữa anchor và positive là nhỏ nhất, trong khi khoảng cách giữa anchor và negative là lớn nhất bằng cách tối ưu hóa hàm mất mát triplet sau:

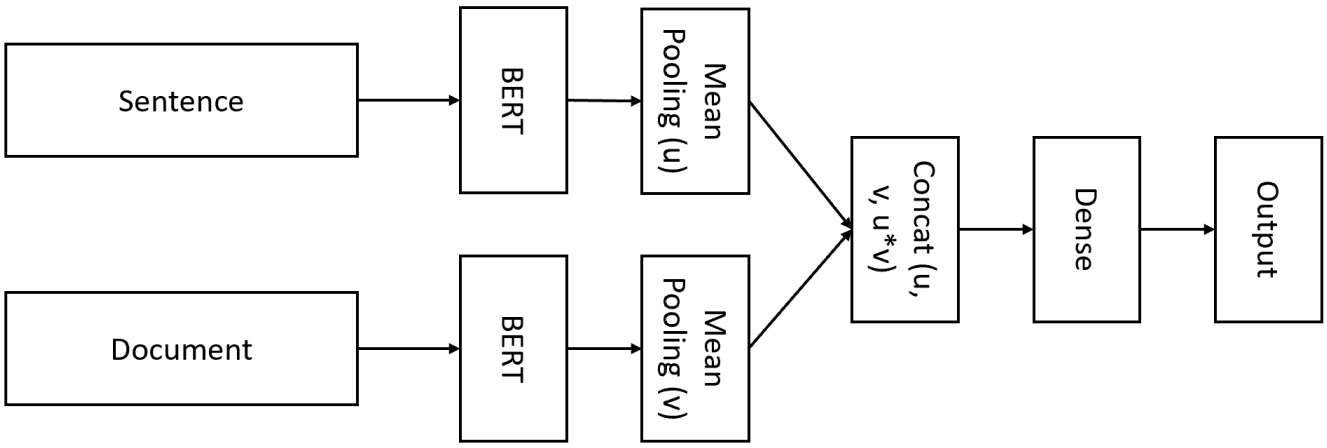
$$max(||s_a - s_p|| - ||s_a - s_n|| + \epsilon, 0)$$

trong đó, s_x vector đặc trưng của "anchor" a, "positive" p và "negative" n, $\epsilon > 0$ là margin, được sử dụng để đảm bảo rằng s_p gần s_a hơn s_n.

3.2 Mô hình đề xuất

Trong báo cáo này, em sử dụng phương pháp tóm tắt trích chọn - extraction nên hướng tiếp cận bài toán là: "chấm điểm" từng câu trong đoạn văn để tính toán khả năng câu đó có nằm trong bản tóm tắt không. Do đó, em sẽ xây dựng một mô hình phân loại với input là một câu và toàn bộ tài liệu, ouput là điểm thể hiện khả năng có thuộc phần tóm tắt hay không.

Dưới đây là mô tả kiến trúc của mô hình trên.



Hình 16: Kiến trúc của mô hình đề xuất

Kiến Trúc Mô Hình

Input của mô hình là câu (sentence) và đoạn văn bản chứa câu đó (document). Cả hai được đưa vào mô hình BERT để tạo các word embedding, tiếp tục được đưa qua lớp mean pooling để tạo sentence embedding.

Các sentence embedding được nối lại với nhau và được xử lý qua một lớp dense trước khi đạt đến lớp output cuối cùng.

Tính toán embedding

Mô hình sử dụng mean pooling để tính toán embedding trung bình cho mỗi câu.

Lớp dense

Là lớp tuyến tính bao gồm lớp pre-classifier, dropout và classifier.

Phân loại câu

Mô hình được sử dụng để phân loại câu để xem xét xem câu đó có nên được bao gồm trong đoạn văn bản tóm tắt hay không. Điều này được thực hiện bằng cách tính toán tích element-wise của các embedding câu và tài liệu, sau đó nối các embedding này với tích của chúng. Kết quả sau cùng được truyền qua một lớp Dense để tạo ra đầu ra cuối cùng của mô hình.

4 Kết quả và đánh giá

4.1 Bộ dữ liệu thử nghiệm

Bộ dữ liệu: em sử dụng bộ dữ liệu CNN daily mail tại đây

Bộ dữ liệu có kích thước 1.29 GiB gồm 3 tập: train, validation và test với số lượng của mỗi tập như sau:

	Train	Validation	Test
Kích thước	287 113	13 368	11 490

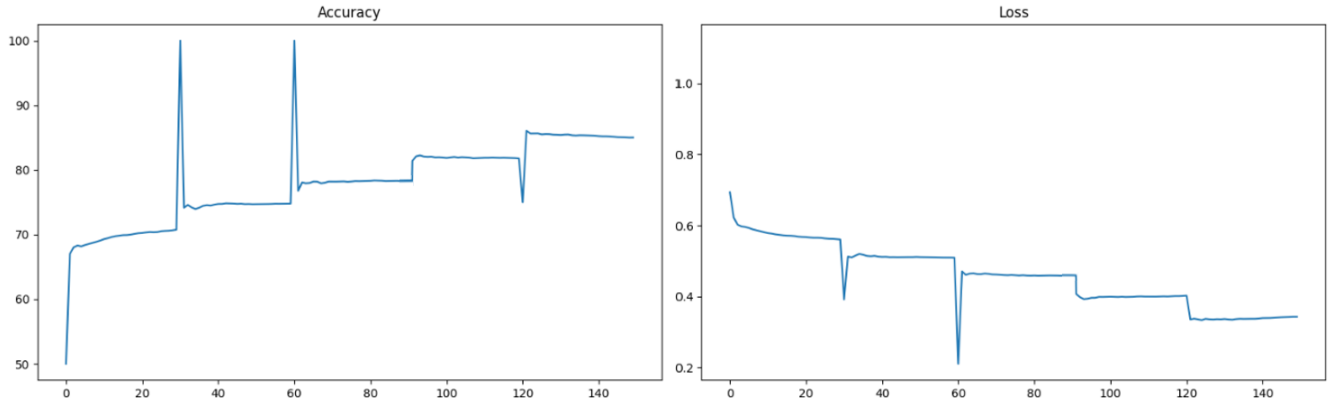
Bộ dữ liệu có 4 thuộc tính:

- "article": nội dung của bài báo, tin tức
- "highlights": bản tóm tắt của "article" (tiêu điểm)
- "id": ID của bài báo
- "publisher": tên nhà xuất bản

4.2 Huấn luyện mô hình

Chúng tôi sử dụng Adam Optimizer với learning rate = 1e-5 để tối thiểu hàm mất mát Binary Cross Entropy Loss (BCELoss).

Dưới đây là biểu diễn của hàm loss và độ chính xác accuracy khi huấn luyện mô hình.



Hình 17: Độ chính xác và giá trị hàm mất mát của mô hình

4.3 Đánh giá độ chính xác

4.3.1 Đánh giá mô hình

Để đánh giá mô hình, em sử dụng hệ thống độ đo ROUGE [6] là hệ thống ước lượng chính. ROUGE - Recall-Oriented Understudy for Gisting Evaluation dùng để xác định chất lượng bản tóm tắt sinh ra từ mô hình với bản tóm tắt mẫu. Mức độ đo lường được tính bởi số lượng đơn vị trùng lặp như N-grams, chuỗi các từ, cặp các từ.

- ROUGE-1: tính toán sự trùng lặp của các unigram (các từ đơn) giữa bản tóm tắt được sinh ra và bản tóm tắt mẫu.
- ROUGE-2: tương tự như như ROUGE-1, thay vì tìm sự trùng lặp của các unigram, ROUGE-2 tìm sự trùng lặp của các bigram (2 từ liên tiếp nhau).

- ROUGE-L: tính toán sự trùng lặp dựa trên chuỗi con dài nhất (Longest Common Subsequence - LCS). Chuỗi con dài nhất không nhất thiết liên tiếp nhưng phải đúng thứ tự.

Cả 3 biến thể trên của ROUGE đều sử dụng các độ đo precision, recall và F1-score để đánh giá:

- Precision: Tỷ lệ giữa số lượng n-gram chung giữa bản tóm tắt được sinh ra và bản tóm tắt mẫu trên tổng số n-gram trong bản tóm tắt được sinh ra. Ví dụ với ROUGE-1:

$$\text{precision} = \frac{\text{số unigram chung}}{\text{Tổng số unigram trong bản tóm tắt được sinh ra}}.$$

- Recall: Tỷ lệ giữa số lượng n-gram chung giữa bản tóm tắt được sinh ra và bản tóm tắt mẫu trên tổng số n-gram trong bản tóm tắt mẫu. Ví dụ với ROUGE-1:

$$\text{precision} = \frac{\text{số unigram chung}}{\text{Tổng số unigram trong bản tóm tắt mẫu}}.$$

- F1-score: Sự kết hợp của precision và recall, được tính bằng công thức:

$$F1 = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}.$$

4.3.2 Kết quả thử nghiệm

Do cấu hình máy huấn luyện hạn chế nên em dừng lại quá trình huấn luyện ở epoch 5 để kiểm tra kết quả.

	ROUGE-1	ROUGE-2	ROUGE-L
Precision	12.303	0.437	11.891
Recall	80.134	43.741	77.493
F1-score	20.801	0.832	20.105

Bảng 1: Đánh giá độ chính xác trên tập gồm 1964 article

5 Kết luận và hướng phát triển

Qua quá trình huấn luyện mô hình SBERT để tóm tắt văn bản, em đã đạt được những kết quả khả quan. Cụ thể, mô hình của em đã đạt độ chính xác lên đến 85% trên tập huấn luyện và 70% trên tập kiểm thử sau 5 epoch. Những kết quả này cho thấy rằng SBERT có tiềm năng lớn trong việc tóm tắt văn bản, mặc dù vẫn còn một số hạn chế cần được khắc phục.

Trong tương lai, em dự định tiếp tục cải thiện mô hình của mình thông qua việc tinh chỉnh các tham số và sử dụng thêm dữ liệu huấn luyện. Em cũng sẽ thử nghiệm với các biến thể khác của SBERT để xem liệu chúng có thể mang lại hiệu suất tốt hơn hay không.

6 Tài liệu tham khảo

Tài liệu

[1] Dzmitry Bahdanau, KyungHyun Cho & Yoshua Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate", *International Conference on Learning Representations, ICLR 2015*, 2015, doi: arXiv.1409.0473.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser & Illia Polosukhin, "Attention Is All You Need", *Neural information processing systems, NIPS2017*, 2017, doi: arXiv.1706.03762.

[3] Colin Raffel & Daniel P. W. Ellis, "Feed-Forward Networks with Attention Can Solve Some Long-Term Memory Problems", *International Conference on Learning Representations, ICLR 2016*, 2016, doi: arXiv.1512.08756.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee & Kristina Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2018, doi: arXiv.1810.04805.

[5] Nils Reimers & Iryna Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks", *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, doi: arXiv.1908.10084.

[6] Chin-Yew Lin, "ROUGE: A Package for Automatic Evaluation of Summaries", *Text Summarization Branches Out*, 2004, doi: arXiv.W04-1013.