Mô hình Ngôn ngữ Yểm mã tối ưu cho Tiếng Việt

Võ Anh Khoa
Công ty Cổ phần Công nghệ OLLI
TP.Hồ Chí Minh, Việt Nam
Email: khoa@olli-ai.com

Đỗ Đức Hào
Công ty Cổ phần Công nghệ OLLI
TP.Hồ Chí Minh, Việt Nam
Email: hao@olli-ai.com

Lâm Quang Tường
Công ty Cổ phần Công nghệ OLLI
TP.Hồ Chí Minh, Việt Nam
Email: tuong@olli-ai.com

Lý Quốc Thắng
Công ty Cổ phần Công nghệ OLLI
TP.Hồ Chí Minh, Việt Nam
Email: thang@olli-ai.com

***Tóm tắt nội dung*—**

***Keywords*-component; formatting; style; styling;**

I. Giới thiệu

Mô hình Ngôn ngữ Yểm mã (Masked Language Model) nói riêng hay Mô hình Ngôn ngữ (Language Model) nói chung là hướng nghiên cứu quan trọng trong chuyên ngành xử lý ngôn ngữ tự nhiên. Cùng với sự phát triển của phương pháp Học sâu (Deep Learning) cùng với các mô hình mạng học sâu dạng Transformer[1] đã cho thấy sự hiệu quả trong việc cải thiện rất nhiều cho các tác vụ xử lý ngôn ngữ tự nhiên [2]. Những tác vụ đó được chia làm hai loại tác vụ mức độ câu (sentence-level tasks) như nhận dạng tính đồng nghĩa (paraphasing) và tác vụ mức độ token (toke-level tasks) như nhận dạng thực thể (named entity recognition).

Cùng với đó, có hai phương pháp áp dụng biểu diễn mô hình ngôn ngữ đã được huấn luyện trước (pre-trained language representation), bao gồm: sử dụng biểu diễn đặc trưng (feature-based) và tinh chỉnh (fine-tuning). Trong đó định hướng sử dụng biểu diễn đặc trưn thì mô hình ngôn ngữ được huấn luyện trước sẽ cung cấp cho vector đặc trưng cho đối tượng cần rút trích (đoạn, câu, cụm từ, từ) nhằm phục vụ cho mục đích luyện Định hướng tinh chỉnh thì mô hình ngôn ngữ lại được dùng nhưng 1 phần tử của để rút trích và tinh chỉnh biểu diễn đặc trưng cho đối tượng cần huấn luyện nhằm thu gọn các tham số của mô hình mà chỉ sử dụng cho các tác vụ đặc định (task-specific). Kể từ khi có sự ra đời các mô hình học mạng học sâu như OpenAI GPT[3], BERT[4], XLNet[5], DistilBERT[6], RoBERTa[7], XLM-R[8], OpenAI GPT2[9], ALBERT[10], OpenAI GPT3[11]... cộng đồng nghiên cứu xử lý ngôn ngữ tự nhiên trên thế giới và trong nước có những chuyển biến mạnh mẽ trong các hoạt động nghiên cứu và ứng dụng mô hình mạng học sâu dạng Transformer có những kết quả tốt hơn đáng kể so với các kết quả sử dụng những phương pháp cũ hơn. Đặc biệt, trong tháng 05/2020 VinAI công bố nghiên cứu về PhoBERT[12] - một mô hình ngôn ngữ yểm mã dành cho tiếng Việt đã cho thấy sức ảnh hưởng của xu hướng trên cùng kết quả cải thiện đáng kể cũng như bắt đầu chứng minh được tính khả thi trong các ứng dụng nghiên cứu trong chuyên ngành xử lý ngôn ngữ tự nhiên có liên quan mà sử dụng mô hình ngôn ngữ yểm mã theo hướng biểu diễn đặc trưng hay theo hướng tinh chỉnh. Tuy nhiên, ngay cả những mô hình được sự hỗ trợ huấn luyện từ tài nguyên tính toán dồi dào như PhoBERT[12] cũng vẫn đang có những hạn chế cố hữu và có thể dễ dàng nhận ra khi sử dụng trong các tác vụ đặc định mà căn bản nguyên nhân chính là sự lựa chọn mô hình, lựa chon ngữ liệu tiếng Việt để huấn luyện cùng với phương pháp huấn luyện và các kĩ thuật tiền xử lý.

Mô hình ngôn ngữ yểm mã sử dụng phương pháp học sâu dạng Transfomer tuy được hình thành và phát triển mạnh mẽ gần đây nhưng bản chất nó được lấy cảm hứng từ nhiệm vụ hoàn hình[13] (Cloze task). Khi đó mô hình ngôn ngữ yểm mã bắt đầu che đi (yểm) một cách ngẫu nhiên các token (mã) và mục tiêu là mạng cần phải đoán ra (hoàn hình) được các token tương ứng trong tập từ điển định trước một cách phù hợp. Chính vì định hướnng này đưa ra một hướng đi mới cho định hướng xử lý ngôn ngữ tự nhiên hiện đại theo một cách tự nhiên hơn và không còn chịu sự ràng buộc quá nhiều tài

nguyên dữ liệu cho huấn luyện về tính gán nhãn nhưng cũng đặt ra những thách thức nhãn tiền cho các nghiên cứu cho sự nghiên cứu một mô hình tinh gọn, đủ tối ưu và đủ tốt cho tiếng Việt ngày nay.

Chúng tôi đề xuất mô hình ... (chưa biết tên gì)... dựa trên những nghiên cứu gần đây về các mô hình hiện đại của Google và Facebook nhưng có sự tinh giãn, chắc lọc những tinh tuý cần thiết theo chúng tôi là phù hợp cho tiếng Việt, phù hợp cho việc huấn luyện mô hình không tốn quá nhiều tài nguyên, tài lực. Đặc biệt hơn, mô hình chứng minh được tính hiệu quả trong quá trình suy lý (inference) được sử dụng như là một phần trong các tác vụ đặc định trong môi trường công nghiệp (industrial environment).

Khác với hai hướng tiếp cận chính hiện nay trong xây dựng của mô hình ngôn ngữ yểm mã cho tiếng việt:

- Đối với hướng tiếp cận hướng mô hình như BERT[4], XLM-R[8], tuy rằng các mô hình này vẫn cho được kết quả tương đối tốt nhưng vẫn chưa hiệu quả khi mà mô hình sử dụng dữ liệu đa ngôn ngữ với số lượng dữ liệu vô cùng lớn (2.5 TB) và ít được tiền xử lý làm cho mô hình trở nên quá nặng nề (số lượng tham số ở mức trên trăm triệu) và tỉ lệ mất cân bằng khi mà tỉ lệ ngữ liệu tiếng Việt chiếm một phần thiểu số trong tương quan các ngôn ngữ khác dựa trên đánh giá tập từ điển được công bố. Mô hình chúng tôi dựa trên khung mô hình của RoBERTa[7] khi mà lược bỏ đi tác vụ đoán câu tiếp theo (Next Sentence Prediction) do bài báo về BERT[4] đề xuất và chỉ sử dụng 8 lớp Transfomer thay vì 12 hay 24 của các mô hình ngôn ngữ vừa nêu trên. Cùng với đó chúng tôi thay thế phương pháp huấn luyện, thuật toán tối ưu cùng phương pháp chọn siêu tham số (hyperparameters) tham khảo từ github của NVIDIA [1] để có thể huấn luyện được mô hình tối ưu nhất trên GPU. Song song với đó lượng ngữ liệu được xử lý và chon lọc từ nguồn ngữ liệu tin tức tiếng Việt [2] của với dung lượng dữ liệu chỉ 10GB.
- Đối với cách tiếp cận của PhoBERT[12], chúng tôi loại bỏ hoàn toàn việc tiền xử lý bằng cách tách các từ trước khi sử dụng trong huấn luyện. Trong thực tế thông qua thực nghiệm thì thuật toán tách từ có những sai số tuy nhỏ nhưng ảnh hưởng đến kết quả cuối cùng của mô hình.
- Mô hình chúng tôi đề xuất là tiên tiến nhất khi so sánh Perplexity trong tất cả các mô hình cũng như tính nhẹ của mô hình khi mà số tham số ít hơn hẳn các mô hình đã nêu trên (dưới một trăm triệu).

[1]https://github.com/NVIDIA/DeepLearningExamples/tree/master/FasterTransformer

[2]https://github.com/binhvq/news-corpus

II. Nền tảng

A. Transformer

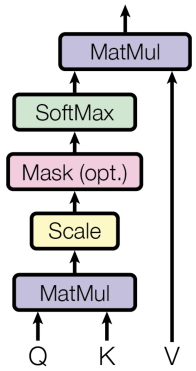
Theo như hình **??** Đi từ bản chất độ attention cho thấy mối liên hệ theo từng cặp token (x, y) với nhau khi đó.

Thông qua ma trận trọng số W_Q, W_K ta tính được q,k (Querry và Key) của x và

Tương tự với ma trận trọng số W_V ta thu được v (value) của y và khi đó.

Để chuẩn hoá ta sử dụng tham số độ dài của chuỗi nhập vào là d_k. Khi đó ta tính dc Attention score (a) như sau:

$$\text{a} = \text{softmax}(\frac{qk^T}{\sqrt{d_k}}v)$$



Hình 1: Attention dùng Tich vô hướng

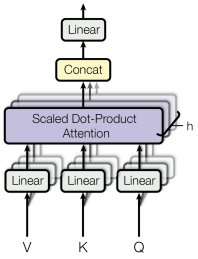
Theo như hình 2 Một cách tương tự cho nhiều ma trận thành 1 tensor ta có Multihead Attention được hiểu là:

Thông qua tensor trọng số W_Q, W_K ta tính được Q,K của x và

Tương tự với tensor trọng số W_V ta thu được V (value) của y và khi đó.

Để chuẩn hoá ta sử dụng tham số độ dài của chuỗi nhập vào là d_k. Khi đó ta tính dc Attention score (A) như sau:

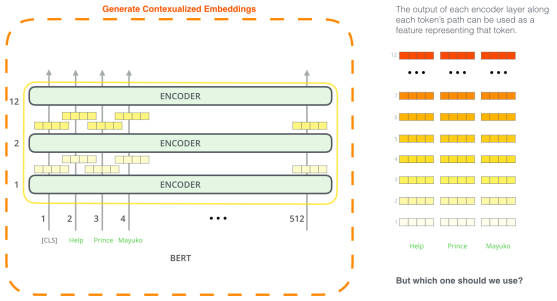
$$\text{a} = \text{softmax}(\frac{QK^T}{\sqrt{d_k}}V)$$



Hình 2: Multi-Head Attention

Mô hình transformers có bản chất là tìm ra tất cả các mối liên hệ token và chính các ma trận trong số lưu giữ các mối liên hệ này như những dạng biểu diễn đặc trưng cho những tính chất nào đó mà câu đó đang mang theo dựa vào ngữ liệu và mục tiêu huấn luyện.

Do đó về bản chất mô hình BERT[4] như hình 3 hay RoBERTa[7] thì là việc chồng chất các quan hệ ẩn dưới dạng các quan hệ được tính bằng attention score và tính chấp nối qua dạng resudial connection như là: LayerNorm$(x + \text{Sublayer}(x))$.



Hình 3: Mô hình Transformers

Nhờ vào tính chồng chất và tính tổ hợp của các chồng chất quan hệ biểu diễn trong không gian ẩn mà khi huấn luyện cho các tác vụ đặc định mà mô hình ngôn ngữ yếm mã nói riêng hay mô hình ngôn ngữ nói chung tỏ ra vượt trội hơn các tiếp cận cũ [12].

B. RoBERTa

RoBERTa[7] là viết tắt của **R**obustly **O**ptimized **BERT** pretraining **A**pproach là một cách tiếp cận mà chúng tôi sử dụng là khung xương cho toàn bộ mô hình chúng tôi.

C. FasterTransformer

Để huấn luyện và sử dụng cho các tác vụ đặc định hiệu quả trên những tài nguyên tính toán dùng cho học sâu như GPU. Chúng tôi nhân thấy việc sử dụng lớp self-attention cho GPU do NVIDIA hỗ trợ phù hợp cho việc tính toán các khối ma trận tối ưu như ở hình 5.

Ở hình 5 chúng ta thấy rõ sự tối ưu nếu như mô hình được tính toán theo khối ma trận hay tensor một cách song song trên GPU giúp tiết kiệm được thời gian tính toán đáng kể.

D. ALBERT

ALBERT[10] hay là **A** **L**ite **BERT** là mô hình tuy có số lượng tham số nhỏ hơn [4] tuy nhiên lại thể hiện kết quả cải thiện đáng kể so với tiền nhiệm. Trong mô hình ngôn ngữ chúng tôi sử dụng hai kĩ thuật sau làm nền tảng để bước đầu thu gọn kích thước mô hình như không làm mất mà con tăng dần tính hiệu quả của mô hình.

1) Phân giải Embedding (Factorized embedding parameterization): Nhằm mục đích giảm nhẹ kích thước mô hình chúng tôi phân giản ma trận tham số embedding thành 2 ma trân nhỏ hơn. Thay cho thực hiện phép chiêu vector one-hot trên không giản ẩn (hidden space) có kích thước là H, chúng tôi sử dụng hướng dẫn trong paper ALBERT[10] là chiếu xuốnng không gian embedding có số chiều là E trước rồi mới chiếu xuống không gian ẩn. Nhờ vậy nếu $H >> E$ thì số tham số embeddinng sẽ giảm từ $O(V \times H)$ xuống $O(V \times E + E \times H)$

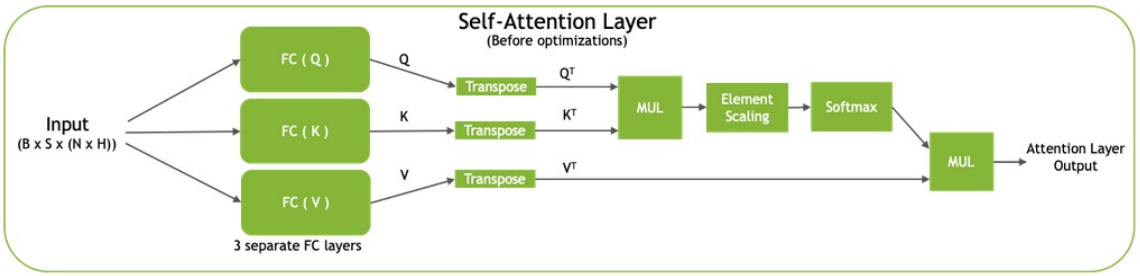
2) Cộng hưởng tham số giữa các lớp (Cross-layer parameter sharing): Nhằm mục đích tăng tốc qua trình huấn luyện chúng tôi cũng sử dụng những kĩ thuật cộng hưởng tham số và lựa chọn cấu hình của kết quả thí nghiệm tốt nhất đề cập trong [10]. Nhờ quá trình cộng hưởng tham số quá trình cập nhật trong số sẽ nhanh hơn và mô hình cũng sẽ càng thêm nhẹ nhàng vì không cần quá nhiều thông số để mà lưu trữ thông tin được phiên mã (encoded) trong mô hình.

III. Dữ liệu và phương pháp nghiên cứu

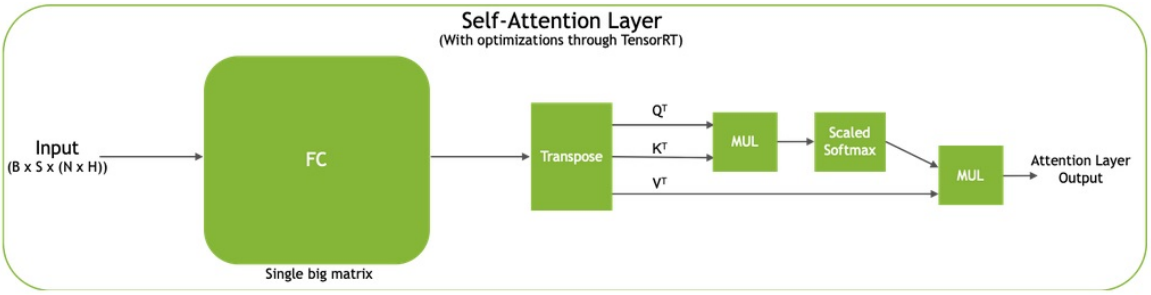
A. Kiến trúc

Chúng tôi xây dựng bốn mô hình thí nghiệm

1) Sử dụng mô hình RoBERTa[7] ở cấu hình base với sự thay đổi là chỉ là 8 lớp Transfomer và độ dài chuỗi đầu vào tối đa 128.
2) Sử dụng mô hình 1 nhưng các lớp Transformers được thay bằng FasterTransformer.
3) Sử dụng mô hình 2 nhưng các lớp lớp Embddining thông thường được thay bằng lớp Embedding được phân giải.



Hình 4: Lớp Self-Attention thông thường [3]



Hình 5: Lớp Self-Attention cho GPU sử dụng trong modoule FasterTransformer [4]

4) Sử dụng mô hình 3 và Công hướng tham số giữa các lớp sau khi tính toán độ attention trong các lớp dựa vào [10].

B. Mục tiêu huấn luyện

Chúng tôi hoàn toàn sử dụng lại hàm lỗi và mục tiêu huấn luyện chỉ cho tác vụ của mô hình ngôn ngữ yểm mã trong bài báo RoBERTa[7].

C. Thuật toán tối ưu

Do mô hình học không huấn luyện trên TPU (Tensor Processing Unit) như cách bài báo ALBERT[10] sử dụng thuật toán LAMB[14], vì thế để có thể huấn luyện tối ưu với batch size lớn (2048-4096) chúng tôi đã sử dụng thuật toán tối ưu tối ưu trên GPU là NVLAMB 1.

D. Dữ liệu

Chúng tôi tiến hành sử dụng ngữ liệu tin tức miễn phí như đã trình bày trong I và tiến hành lọc và tách các từ theo thuật toán2:

Trong quá trình thực nghiệm nhiều lần, chúng tôi nhận ra việc tiền xử lý có vai trò dần quan trọng và không thể bỏ đi dù cho mô hình học sâu có tiên tiến đến đâu. Chúng tôi đang nghiên cứu và cải thiện dần thuật toán tạo bộ ngữ liệu chuẩn phù hợp nhất các yếu tố ngôn ngữ hoc của tiếng Việt.

E. Quá huấn luyện

IV. Thí nghiệm và Kết quả

Chúng tôi thực nghiệm huấn luyện mô hình trên cấu hình máy tính với 16 CPU Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz và 4 GPU NVIDIA Tesla V100 - 32GB VRAM. Framework dùng để huấn luyện là Pytorch sử dụng một phần mã nguồn của HuggingFace-Transformers[13] và NVIDIA-DeepLearningExamples[6]. Thời gian huấn luyện mô hình trung bình là 12 ngày. Chúng tôi sử dụng ngữ liệu huấn luyện và kiểm thử theo những dữ liệu công khai đã được trình bày trong III-D. Kết quả thực nghiệm nhiều mô hình ngôn ngữ có hỗ trợ tiếng việt được trình bày ở Bảng I

V. Kết luận

Cảm tạ

Tài liệu

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need. arxiv 2017," *arXiv preprint arXiv:1706.03762*, 2017.

[2] A. M. Dai and Q. V. Le, "Semi-supervised sequence learning," in *NIPS*, 2015.

[6]https://github.com/NVIDIA/DeepLearningExamples/

Algorithm 1: Thuật toán tối ưu NVLAMB dùng cho GPU

Result: Trọng số mô hình tại bước $t+1$

Khởi tạo: Tại bước t với mini-batch x và trọng số mô hình là $w(t)_l^i$

Bước 1: Tính đạo hàm $g(t)_l^i$của trọng số mô hình $w(t)_l^i$

Bước 2: Chuẩn hoá đạo hàm:

$$g(t) = \forall_{l=1}^{L} \text{concatentate}(g(t)_l)$$
$$g_{norm}(t) = \|g(t)\|_2$$

if $g_{norm}(t) > 1$ **then**

$\quad \widehat{g}(t)_l^i = \frac{g(t)_l^i}{g_{norm}(t)}$

else

$\quad g(t)_l^i$

end

Bước 3: Cập nhật hàm momentum $m(t)$, hàm vận tốc $v(t)$ tương ứng với trọng số của lớp l là $w(t)_l^i$ dựa trên đạo hàm $g(t)$ và các siêu tham số β_1 và β_2:

$$m(t)_l^i = \beta_1 m(t-1)_l^i + (1-\beta_1)\,\widehat{g}(t)_l^i \quad (1)$$

$$v(t)_l^i = \beta_2 m(t-1)_l^i + (1-\beta_2)\,(\widehat{g}(t)_l^i)^2 \quad (2)$$

Bước 4: Áp dụng beta-correction:

$$\widehat{m}(t)_l^i = \frac{m(t)_l^i}{\left(1-(\beta_1)^t\right)} \quad (3)$$

$$\widehat{v}(t)_l^i = \frac{v(t)_l^i}{\left(1-(\beta_2)^t\right)} \quad (4)$$

Bước 5: Cập nhật $u(t)_l^i$ dựa vào trọng số $w(t)_l^i$ dựa vào tham số λ và ϵ:

$$u(t)_l^i = \frac{\widehat{m}(t)_l^i}{\sqrt{\widehat{v}(t)_l^i + \epsilon}} + \lambda w(t)_l^i$$

Bước 6: Tính tỉ lệ $r(t)_l$ giữa trọng số $w(t)_l$ và độ lớn của tham số cập nhật $u(t)_l$:

$$r(t)_l = \frac{\|w(t)_l\|_2}{\|u(t)_l\|_2}$$

Bước 7: Cập nhật trọng số $w(t)_l$ và tốc độ học (learning rate) lr:

$$w(t+1)_l^i = w(t)_l^i - lr \times r(t)_l \times u(t)_l^i$$

.

Algorithm 2: Thuật toán khai thác và tách từ để tạo ngữ liệu cho huấn luyện mô hình ngôn ngữ.

Result: Ngữ liệu và từ điển cho huấn luyện mô hình ngôn ngữ.

Khởi tạo: Ngữ liệu huấn luyện theo và kiểm thử đã nêu ở I.

Bước 1: Lược bỏ các dòng trùng lặp.

Bước 2: Kiểm tra tính hợp lệ của câu tiếng Việt theo thuật toán Trích lọc tiếng Việt [5].

Bước 3: Tách ngữ liệu thành dạng mỗi dòng một câu.

Bước 4: Tạo từ điển và tách câu thành dạng token-id vector bằng thuật toán byte-pair-encoding (BPE)[15].

Bước 5: Lựa chọn các câu mà số phần tử token-id vector nhỏ hơn 128.

[3] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding with unsupervised learning," *Technical report, OpenAI*, 2018.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[5] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," in *Advances in neural information processing systems*, 2019, pp. 5753–5763.

[6] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[7] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[8] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, "Unsupervised cross-lingual representation learning at scale," *arXiv preprint arXiv:1911.02116*, 2019.

[9] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.

[10] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," *arXiv preprint arXiv:1909.11942*, 2019.

ID	Language Model	Parameters	Perplexity
1	xlm-roberta-base	278.89M	495.39
2	xlm-roberta-large	561.19M	13.16
3	bert-base-multilingual-uncased	168.05M	167.2
4	bert-base-multilingual-cased	178.57M	78.77
5	distilbert-base-multilingual-cased	135.45M	454.15
6	pho-bert-base	135.65M	81.74
7	pho-bert-large	370.27M	65.19
8	vietnamese-roberta-cased	37.68M	99.24
9	vietnamese-roberta-uncased	34.58M	138.44
10	albert-vi	18.11M	560.08
11	roberta-base (ours)	111.04M	31.24
12	roberta-base (FasterTransformer) (ours)	92.59M	24.97
13	roberta-base (FasterTransformer) + cross-layer (ours)	80.82M	16.30
14	roberta-base (FasterTransformer) + cross-layer + factorized embedding (ours)	76.79M	**12.11**

Bảng I: Kết quả thực nghiệm trên tập kiểm thử

[11] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[12] D. Q. Nguyen and A. T. Nguyen, "Phobert: Pre-trained language models for vietnamese," *arXiv preprint arXiv:2003.00744*, 2020.

[13] W. L. Taylor, ""cloze procedure": A new tool for measuring readability," *Journalism quarterly*, vol. 30, no. 4, pp. 415–433, 1953.

[14] Y. You, J. Li, S. Reddi, J. Hseu, S. Kumar, S. Bhojanapalli, X. Song, J. Demmel, K. Keutzer, and C.-J. Hsieh, "Large batch optimization for deep learning: Training bert in 76 minutes," *arXiv preprint arXiv:1904.00962*, 2019.

[15] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," *arXiv preprint arXiv:1508.07909*, 2015.