**PhoBERT: Mô hình ngôn ngữ tiền huấn luyện cho Tiếng Việt**

Nguyễn Quốc Đạt, Nguyễn Tuấn Anh - VinAI Research, Việt Nam; NVIDIA, USA Nơi phát hành: Hội nghị IJCAI[[1]](#footnote-2) năm 2020.

# Abstract

Chúng tôi giới thiệu PhoBERT với hai phiên bản - PhoBERTbase và PhoBERTlarge[[2]](#footnote-3). Đây là mô hình đơn ngữ[[3]](#footnote-4) quy mô lớn đầu tiên được tiền huấn luyện[[4]](#footnote-5) cho tiếng Việt. Kết quả thử nghiệm cho thấy rằng PhoBERT luôn vượt trội hơn so với mô hình đa ngôn ngữ[[5]](#footnote-6) tiền huấn luyện tốt nhất hiện tại XLM-R[[6]](#footnote-7) (Conneau et al., 2020) và cải thiện (state-of-the-art[[7]](#footnote-8)) trong nhiều tác vụ xử lý ngôn ngữ tự nhiên dành riêng cho tiếng Việt[[8]](#footnote-9), bao gồm gán nhãn từ loại, phân tích cú pháp phụ thuộc, trích xuất thông tin thực thể và suy luận ngôn ngữ tự nhiên. Chúng tôi phát hành PhoBERT để hỗ trợ cho nghiên cứu và các ứng dụng tiếp theo trong lĩnh vực xử lý ngôn ngữ tự nhiên cho tiếng Việt. Các mô hình PhoBERT của chúng tôi có sẵn tại: <https://github.com/VinAIResearch/PhoBERT>.

# 1 Introduction

Mô hình tiền huấn luyện ngôn ngữ, đặc biệt là BERT[[9]](#footnote-10) (Devlin et al., 2019) - Biểu diễn trình mã hóa hai chiều từ Transformers (Vaswani et al., 2017), đã trở nên rất phổ biến và giúp cải thiện đáng kể cho các tác vụ NLP khác nhau. Thành công của BERT trong tiền huấn luyện và các biến thể của nó phần lớn nhưng chỉ giới hạn ở ngôn ngữ là tiếng Anh. Đối với các ngôn ngữ khác, người ta có thể đào tạo lại một mô hình dành riêng cho ngôn ngữ đó bằng cách sử dụng kiến trúc BERT (Cui et al., 2019; de Vries et al., 2019; Vu et al., 2019; Martin et al., 2020) hoặc sử dụng các mô hình dựa trên BERT tiền huấn luyện đa ngôn ngữ đã có sẵn (Devlin et al., 2019; Conneau and Lample, 2019; Conneau et al., 2020).

Đối với việc mô hình hóa ngôn ngữ tiếng Việt, theo sự hiểu biết tốt nhất của chúng tôi, có hai vấn đề chính cần quan tâm như sau:

* Bộ dữ liệu Wikipedia là bộ dữ liệu duy nhất được sử dụng để huấn luyện các mô hình đơn ngữ (Vu et al., 2019), và đây cũng là bộ dữ liệu tiếng việt duy nhất được sử dụng bởi tất cả mô hình đa ngôn ngữ ngoại trừ XLM-R. Điều cần chú ý là dữ liệu từ Wikipedia không đại diện cho việc sử dụng ngôn ngữ thông thường[[10]](#footnote-11) và dữ liệu Wikipedia Tiếng Việt có kích thước tương đối nhỏ (1GB sau khi giải nén), trong khi các mô hình ngôn ngữ tiền huấn luyện có thể được cải thiện đáng kể bằng cách sử dụng nhiều dữ liệu tiền huấn luyện hơn (Liu et al., 2019).
* Tất cả mô hình ngôn ngữ BERT đơn ngữ và đa ngôn ngữ được phát hành công khai đều không phân biệt được sự khác biệt giữa các âm tiết và các từ[[11]](#footnote-12) trong tiếng Việt. Sự mơ hồ này xuất phát từ việc khoảng trống cũng được sử dụng để phân tách các âm tiết tạo thành từ khi viết bằng tiếng Việt[[12]](#footnote-13). Ví dụ, một đoạn văn viết gồm 6 âm tiết "Tôi là một nghiên cứu viên" tạo thành 4 từ "Tôi I", "là am", "một a", "nghiên\_cứu\_viên researcher". Nếu không tiến hành phân đoạn từ tiếng Việt trước đó, những mô hình này sẽ áp dụng trực tiếp phương pháp mã hóa Byte-Pair (BPE)[[13]](#footnote-14) (Sennrich et al., 2016; Kudo và Richardson, 2018) vào dữ liệu tiền huấn luyện tiếng Việt ở mức âm tiết. Dễ hiểu rằng đối với các tác vụ NLP tiếng Việt cấp từ, những mô hình tiền huấn luyện trên dữ liệu ở mức âm tiết có thể không hoạt động tốt bằng những mô hình tiền huấn luyện trên dữ liệu ở mức từ.

Để giải quyết hai vấn đề trên, chúng tôi tiến hành huấn luyện các mô hình “base” và “large” đầu tiên sử dụng BERT đơn ngữ quy mô lớn với bộ dữ liệu Tiếng Việt có kích thước 20 GB được chia thành cấp từ (ở mức từ). Chúng tôi đánh giá các mô hình của mình dựa trên 4 tác vụ xử lý ngôn ngữ tự nhiên trong Tiếng Việt bao gồm: nhận dạng phần từ (POS), phân tích phụ thuộc và nhận dạng thực thể được đặt tên (NER) ở cấp từ, và một tác vụ hiểu ngôn ngữ tự nhiên (NLI) có thể được sắp xếp dưới dạng tác vụ cấp từ hoặc cấp từng từ. Kết quả thực nghiệm cho thấy các mô hình của chúng tôi đạt được kết quả tốt nhất hiện nay trên tất cả các tác vụ này (Dựa trên SOTA). Các đóng góp của chúng tôi được tóm tắt như sau:

* Chúng tôi trình bày những mô hình đơn ngữ quy mô lớn đầu tiên được tiền huấn luyện cho Tiếng Việt.
* Các mô hình của chúng tôi đạt được kết quả tối ưu (SOTA) trên bốn tác vụ: gán nhãn từ (POS), phân tích phụ thuộc (Dependency parsing), nhận diện thực thể đặt tên (NER) và suy luận ngôn ngữ tự nhiên (NLI), cho thấy tính hiệu quả của các mô hình ngôn ngữ đơn ngữ quy mô lớn dựa trên BERT cho tiếng Việt.
* Theo hiểu biết tốt nhất của chúng tôi, chúng tôi cũng thực hiện bộ thử nghiệm đầu tiên để so sánh các mô hình ngôn ngữ đơn ngữ với mô hình đa ngôn ngữ tốt nhất hiện nay XLM-R trên nhiều tác vụ khác nhau (4 tác vụ kể trên) đối với ngôn ngữ cụ thể (ở đây là tiếng Việt). Các thử nghiệm cho thấy rằng các mô hình của chúng tôi vượt qua XLM-R trên tất cả các tác vụ này, khẳng định một cách thuyết phục rằng các mô hình chuyên sâu riêng cho ngôn ngữ cụ thể vẫn vượt trội hơn so với các mô hình đa ngôn ngữ.
* Chúng tôi công khai phát hành các mô hình của mình dưới tên PhoBERT, có thể sử dụng với fairseq[[14]](#footnote-15) (Ott et al., 2019) và transformers[[15]](#footnote-16) (Wolf et al., 2019). Chúng tôi hy vọng rằng PhoBERT có thể phục vụ như một cơ sở vững chắc cho nghiên cứu và ứng dụng xử lý ngôn ngữ tự nhiên cho tiếng Việt trong tương lai.

# 2 PhoBERT

Phần này trình bày kiến trúc của mô hình và mô tả dữ liệu và quá trình tối ưu hóa được sử dụng để huấn luyện PhoBERT.

**Kiến trúc**: PhoBERT bao gồm hai phiên bản, PhoBERTbase và PhoBERTlarge, sử dụng kiến trúc tương tự với BERTbase và BERTlarge. Phương pháp tiền huấn luyện của PhoBERT được dựa trên RoBERTa[[16]](#footnote-17) (Liu et al., 2019) – mô hình tối ưu hóa quy trình tiền huấn luyện của BERT để có hiệu suất mạnh mẽ hơn.

**Dữ liệu tiền huấn luyện**: Để xử lý vấn đề đầu tiên được đề cập ở Phần 1, chúng tôi sử dụng một bộ dữ liệu tiền huấn luyện gồm khoảng 20GB văn bản không nén. Bộ dữ liệu này là sự kết hợp của hai tập dữ liệu: (i) tập dữ liệu Wikipedia tiếng Việt (khoảng 1GB), và (ii) tập dữ liệu thứ hai (khoảng 19GB) được tạo ra bằng cách loại bỏ các bài viết tương tự và trùng lặp từ một tập dữ liệu tin tức tiếng Việt khoảng 50GB.

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **#training** | **#valid** | **#test** |
| POS tagging*y* Dep. parsing*y* NER*y* NLI*z* | 27,000  8,977  14,861  392,702 | 870  200  2,000  2,490 | 2,120 1,020 2,831 5,010 |

Bảng 0.1. Thống kê các tập dữ liệu dựa trên những tác vụ kể trên. "#training", "#valid" và "#test" đề cập đến kích thước của tập huấn luyện, tập đánh giá và tập kiểm tra, tương ứng. y và z đề cập đến kích thước của tập dữ liệu dưới dạng số câu và số cặp câu, tương ứng.

Để giải quyết vấn đề thứ hai, chúng tôi sử dụng RDRSegmenter [[17]](#footnote-18)(Nguyen et al., 2018) từ VnCoreNLP[[18]](#footnote-19) (Vu et al., 2018) để thực hiện phân đoạn từ và câu trên tập dữ liệu tiền huấn luyện, kết quả cho thấy khoảng 145 triệu câu đã được phân đoạn từ (∼3 tỉ token từ). Khác với RoBERTa, sau đó chúng tôi sử dụng fastBPE[[19]](#footnote-20) (Sennrich et al., 2016) để phân đoạn các câu này thành các đơn vị từ phụ (subword units)[[20]](#footnote-21), sử dụng bộ từ vựng 64K loại con từ. Trung bình có 24,4 đơn vị từ phụ mỗi câu.

**Tối ưu hóa[[21]](#footnote-22)**: Chúng tôi sử dụng phiên bản RoBERTa trong fairseq (Ott et al., 2019). Chúng tôi đặt độ dài tối đa là 256 đơn vị con từ (subword tokens), do đó tạo ra khoảng 145 triệu × 24,4 / 256 ≈ 13,8 triệu khối câu. Theo Liu et al. (2019), chúng tôi tối ưu hóa các mô hình bằng Adam[[22]](#footnote-23) (Kingma và Ba, 2014). Chúng tôi sử dụng kích thước lô là 1024 trên 4 GPU V100 (mỗi GPU có 16GB) và tốc độ học tối đa là 0.0004 cho PhoBERTbase, và kích thước lô là 512 và tốc độ học tối đa là 0.0002 cho PhoBERTlarge. Chúng tôi chạy trong 40 epoch[[23]](#footnote-24) (ở đây, tốc độ học được “khởi động” trong 2 epoch), dẫn đến khoảng 13,8 triệu × 40 / 1024 ≈ 540K bước huấn luyện cho PhoBERTbase và 1,08 triệu bước huấn luyện cho PhoBERTlarge. Chúng tôi tiền huấn luyện PhoBERTbase trong 3 tuần, sau đó là tiền huấn luyện PhoBERTlarge trong 5 tuần[[24]](#footnote-25).

# 3 Experimental setup

Chúng tôi đánh giá hiệu suất của PhoBERT trên bốn tác vụ NLP tiếng Việt sau: Gán nhãn từ loại (POS tagging), phân tích phụ thuộc (Depedency parsing), trích xuất thông tin thực tể (hoặc gọi là gán nhãn thực thể) (NER) và suy luận ngôn ngữ tự liên (NLI).

## Downstream task datasets[[25]](#footnote-26)

Bảng 1 trình bày thống kê các bộ dữ liệu thực nghiệm mà chúng tôi sử dụng để đánh giá tác vụ downstream. Đối với việc đánh dấu phân loại từ loại, phân tích phụ thuộc và gán nhãn thực thể, chúng tôi tuân theo thiết lập VnCoreNLP (Vu et al., 2018), sử dụng các bài kiểm tra tiêu chuẩn của tập dữ liệu đánh dấu phân loại từ loại (POS tagging) VLSP 2013[[26]](#footnote-27), bộ dữ liệu cây phụ thuộc VnDT v1.1[[27]](#footnote-28) (Nguyễn et al., 2014b) với các nhãn từ được dự đoán bởi VnCoreNLP và tập dữ liệu nhận dạng thực thể (NER) VLSP 2016[[28]](#footnote-29) (Nguyễn et al., 2019a).

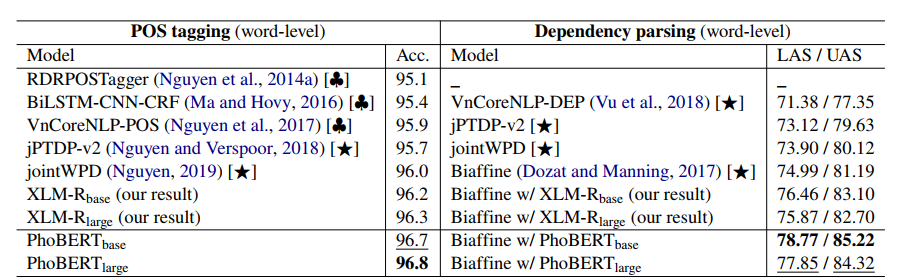
Đối với NLI, chúng tôi sử dụng các tập dữ liệu xác thực (validation) và kiểm tra (test set) tiếng Việt được xây dựng thủ công từ tập dữ liệu đa ngôn ngữ NLI (XNLI) corpus v1.0[[29]](#footnote-30) (Conneau et al., 2018), trong đó tập huấn luyện tiếng Việt được phát hành dưới dạng phiên bản được dịch máy của tập huấn luyện tiếng Anh tương ứng (Williams et al., 2018). Khác với các tập dữ liệu POS tagging, Dependency parsing và NER cung cấp phân đoạn từ vàng (Nghĩa là các tập dữ liệu đã được chia thành các từ đơn lẻ và đã được gán nhãn từ trước), đối với NLI, chúng tôi sử dụng RDRSegmenter để phân đoạn văn bản thành các từ trước khi áp dụng BPE để tạo ra các subword từ các token từ.

## Fine-tuning[[30]](#footnote-31)

Theo Devlin et al. (2019), đối với POS tagging và NER, chúng tôi thêm một lớp dự đoán tuyến tính lên trên kiến trúc PhoBERT (tức là lớp Transformer cuối cùng của PhoBERT) đối với subword đầu tiên của mỗi token từ[[31]](#footnote-32). Đối với phân tích phụ thuộc (depedency parsing), theo Nguyen (2019), chúng tôi sử dụng lại trình phân tích phụ thuộc Biaffine[[32]](#footnote-33) (Phương pháp đang được coi là tốt nhất) (Dozat và Manning, 2017) từ Ma et al. (2018) với các hyperparameter tối ưu mặc định. Sau đó, chúng tôi mở rộng trình phân tích này bằng cách thay thế embedding[[33]](#footnote-34) từ được huấn luyện trước đó của mỗi từ trong câu đầu vào bằng embedding được bao hàm (contextualized embedding[[34]](#footnote-35)) tương ứng (từ lớp cuối cùng) được tính cho subword đầu tiên của từ.

Đối với POS tagging, NER và NLI, chúng tôi sử dụng transformers[[35]](#footnote-36) (Wolf et al., 2019) để fine-tune PhoBERT cho mỗi tác vụ và từng bộ dữ liệu độc lập. Chúng tôi sử dụng AdamW[[36]](#footnote-37) (Loshchilov và Hutter, 2019) với một tốc độ học cố định là 1.e-5 và một kích thước batch là 32 (Liu et al., 2019). Chúng tôi fine-tune trong 30 epochs đào tạo, đánh giá hiệu suất tác vụ sau mỗi epoch trên tập validation (ở đây, early stopping được áp dụng khi không có cải thiện sau 5 epoch liên tiếp), và sau đó chọn điểm kiểm tra tốt nhất để báo cáo kết quả cuối cùng trên tập test (lưu ý rằng mỗi điểm số của chúng tôi là trung bình của 5 lần chạy với các seed ngẫu nhiên khác nhau).

# 4 Experimental results



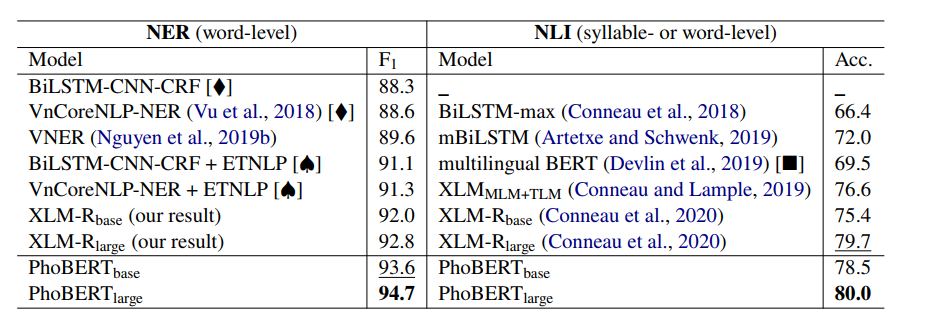
Bảng 2. Điểm số hiệu suất (tính bằng phần trăm) trên tập kiểm tra POS tagging, Dependency parsing. "Acc.", "LAS" và "UAS" lần lượt viết tắt cho Độ chính xác, Điểm số gắn nhãn và Điểm số không gắn nhãn (trong đó, tất cả các số liệu đánh giá này được tính trên tất cả các từ, bao gồm cả dấu câu). [♣] và [\*sao] đề cập đến kết quả được báo cáo bởi Nguyen et al. (2017) và Nguyen (2019).

## Main results

Bảng 2 và bảng 3 so sánh các điểm số của PhoBERT với các kết quả được báo cáo cao nhất trước đây, sử dụng cùng cấu hình thực nghiệm. Rõ ràng rằng PhoBERT của chúng tôi giúp tạo ra các kết quả tốt nhất hiện nay cho tất cả bốn tác vụ phụ thuộc.

Đối với **POS tagging,** (gán nhãn phân loại từ loại), mô hình nơ-ron jointWPD[[37]](#footnote-38) được sử dụng gán nhãn phân loại từ loại và phân tích phụ thuộc chung (Nguyen, 2019) và mô hình dựa trên đặc trưng VnCoreNLP-POS[[38]](#footnote-39) (Nguyen et al., 2017) là hai mô hình SOTA trước đó, đạt độ chính xác khoảng 96,0%. PhoBERT đạt được độ chính xác tuyệt đối cao hơn 0,8% so với hai mô hình này.

Đối với **Depedency parsing** (phân tích phụ thuộc), các điểm số phân tích phụ thuộc LAS[[39]](#footnote-40) và UAS[[40]](#footnote-41) cao nhất trước đó được đạt bởi trình phân tích Biaffine với tỷ lệ lần lượt là 75,0% và 81,2%. PhoBERT giúp tăng cường trình phân tích Biaffine với cải thiện tuyệt đối khoảng 4%, đạt được LAS tại 78,8% và UAS tại 85,2%.



Bảng 3. Điểm số hiệu suất (tính bằng %) trên tập kiểm tra NER và NLI. [♦], [♠] và [] thể hiện kết quả được báo cáo bởi Vu et al. (2018), Vu et al. (2019) và Wu và Dredze (2019). Lưu ý rằng có kết quả NLI tiếng Việt cao hơn ở XLM-R khi fine-tuning trên việc nối tất cả 15 bộ dữ liệu huấn luyện từ tập dữ liệu XNLI (TRANSLATE-TRAIN-ALL: 79,5% cho XLM-Rbase và 83,4% cho XLM-Rlarge). Tuy nhiên, các kết quả đó có thể không so sánh được vì chúng tôi chỉ sử dụng dữ liệu huấn luyện tiếng Việt đơn ngữ để fine-tuning.

Đối với **NER**, PhoBERTlarge có độ F1 cao hơn 1,1 điểm so với PhoBERTbase. Bên cạnh đó, PhoBERTbase cao hơn 2+ điểm so với các mô hình dựa trên tính năng và mạng nơ-ron trước đó như VnCoreNLP-NER (Vu et al., 2018) và BiLSTM-CNN-CRF[[41]](#footnote-42) (Ma và Hovy, 2016) được huấn luyện với tập từ 15K word embeddings dựa trên BERT của ETNLP[[42]](#footnote-43) (Vu et al., 2019).

Đối với **NLI**, PhoBERT vượt trội hơn so với BERT đa ngôn ngữ (Devlin et al., 2019) và mô hình đa ngôn ngữ dựa trên BERT (ở đây là mô hình ngôn ngữ dịch thuật mới là XLMMLM+TLM[[43]](#footnote-44) (Conneau và Lample, 2019)) với khoảng cách rất lớn. PhoBERT cũng vượt trội hơn so với mô hình đa ngôn ngữ được tiền huấn luyện tốt nhất hiện tại là XLM-R nhưng sử dụng số lượng tham số ít hơn rất nhiều so với XLM-R: 135 triệu (PhoBERTbase) so với 250 triệu (XLM-Rbase); 370 triệu (PhoBERTlarge) so với 560 triệu (XLM-Rlarge).

## Discussion

Chúng tôi nhận thấy rằng tại tác vụ Depedency Parsing, PhoBERTlarge đạt được số điểm thấp hơn 0,9% so với PhoBERTbase. Nguyên nhân có thể do lớp Transformer cuối cùng trong kiến trúc BERT không phải là lớp tối ưu nhất có thể mã hóa cấu trúc cú pháp của thông tin phong phú nhất (Có thể hiểu là thông tin phức tạp nhất)(Hewitt và Manning, 2019; Jawahar và đồng nghiệp, 2019). Công việc trong tương lai sẽ nghiên cứu xem lớp Transformer nào của PhoBERT chứa thông tin cú pháp phong phú hơn bằng cách đánh giá hiệu suất phân tích cú pháp tiếng Việt từ mỗi lớp.

Sử dụng nhiều dữ liệu tiền huấn luyện có thể cải thiện đáng kể chất lượng của các mô hình ngôn ngữ tiền huấn luyện (Liu và đồng nghiệp, 2019). Do đó, không ngạc nhiên khi PhoBERT giúp tạo ra hiệu suất tốt hơn so với ETNLP trên NER và các mô hình BERT đa ngôn ngữ và XLMMLM+TLM trên NLI (ở đây, PhoBERT sử dụng 20GB văn bản tiếng Việt trong khi các mô hình đó sử dụng bộ dữ liệu Wikipedia tiếng Việt 1GB).

Sau khi sử dụng phương pháp fine-tuning cho PhoBERT, chúng tôi đã fine-tune cẩn thận XLM-R cho các tác vụ tiếng Việt còn lại là POS tagging, Dependency parsing và NER (ở đây, việc tinh chỉnh được áp dụng vào token phụ âm đầu tiên của âm tiết đầu tiên của mỗi từ)[[44]](#footnote-45). Bảng 2 và 3 cho thấy rằng PhoBERT của chúng tôi cũng làm tốt hơn XLM-R trên ba tác vụ cấp từ này. Đáng chú ý rằng XLM-R sử dụng một bộ dữ liệu tiền huấn luyện có dung lượng 2,5TB chứa khoảng 137GB văn bản tiếng Việt (tức là khoảng 137/20 ≈ 7 lần lớn hơn bộ dữ liệu tiền huấn luyện của chúng tôi). Nhớ lại rằng PhoBERT thực hiện phân đoạn từ tiếng Việt để phân đoạn câu theo âm tiết thành các token trước khi áp dụng BPE để phân đoạn các câu đã phân đoạn từ thành các đơn vị con từ (subword units), trong khi XLM-R trực tiếp áp dụng BPE vào các câu tiếng Việt đã phân đoạn âm tiết. Điều này xác nhận lại rằng các mô hình dành riêng cho một ngôn ngữ vẫn có hiệu suất tốt hơn so với các mô hình đa ngôn ngữ (Martin et al., 2020).

# 5 Conclussion

Trong bài báo này, chúng tôi đã giới thiệu mô hình đơn ngữ quy mô lớn đầu tiên được tiền huấn luyện cho tiếng Việt - PhoBERT. Chúng tôi đã chứng minh tính hữu dụng của PhoBERT bằng cách cho thấy PhoBERT hoạt động tốt hơn so với mô hình đa ngôn ngữ tốt nhất hiện nay là XLM-R và giúp tạo ra kết quả tốt nhất cho bốn tác vụ xử lý ngôn ngữ tự nhiên tiếng Việt bao gồm phân loại từ loại, phân tích phụ thuộc câu, trích xuất thông tin thực thể và suy luận ngôn ngữ tự nhiên. Bằng việc công khai phát hành mô hình PhoBERT, chúng tôi hy vọng chúng có thể thúc đẩy nghiên cứu và ứng dụng trong lĩnh vực xử lý ngôn ngữ tự nhiên tiếng Việt.

1. International Joint Conference on Artificial Intelligence - Hội nghị lớn về Trí tuệ nhân tạo quốc tế được tổ chức hàng năm bởi Hội đồng Nghiên cứu Trí tuệ nhân tạo (AAAI) và Hội đồng Nghiên cứu Máy tính (ACM). [↑](#footnote-ref-2)
2. **PhoBERTbas**e: 12 transformer blocks, **PhoBERTlarge**: 24 transformer block

   Khái niệm transformer block: mục 2-Abstract tài liệu Related Knowledge [↑](#footnote-ref-3)
3. Mô hình đơn ngữ (monolingual model) là một loại mô hình ngôn ngữ sử dụng dữ liệu chỉ của một ngôn ngữ duy nhất để huấn luyện và dự đoán. [↑](#footnote-ref-4)
4. Tiền huấn luyện (pre-trained): Là quá trình huấn luyện mô hình ngôn ngữ trên một lượng lớn dữ liệu tiền xử lý trước đó để học các đặc trưng ngôn ngữ tổng quát, có thể được sử dụng cho nhiều tác vụ khác nhau. Cụ thể tại mục 1-Abstract tài liệu Related Knowledge [↑](#footnote-ref-5)
5. Mô hình đa ngôn ngữ (multilingual model) sử dụng dữ liệu từ nhiều ngôn ngữ để huấn luyện và có khả năng dự đoán cho nhiều ngôn ngữ. [↑](#footnote-ref-6)
6. Mục 3 - Abstract tài liệu Related Knowledge [↑](#footnote-ref-7)
7. **State-of-the-art** là mức độ phát triển cao nhất của một công nghệ, một lĩnh vực khoa học, hoặc một thiết kế nào đó đạt được trong một khoảng thời gian nhất định. [↑](#footnote-ref-8)
8. VietNamese-specific NLP task – mục 4 - Abstract tài liệu Related Knowledge [↑](#footnote-ref-9)
9. BERT: Mô hình hiểu được ngữ cảnh và tương quan giữa các từ trong câu và đưa ra các dự đoán chính xác cho các tác vụ liên quan đến ngôn ngữ như phân loại văn bản, dịch máy, gợi ý từ khóa và phân tích cảm xúc. Cụ thể tại mục 1 – Introduction tài liệu Related Knowledge [↑](#footnote-ref-10)
10. Nghĩa là dữ liệu được thu thập từ Wikipedia không phản ánh chính xác cách sử dụng ngôn ngữ thông thường của người dùng. Nó chỉ đại diện cho một phần của ngôn ngữ được sử dụng trên Wikipedia và có thể không phù hợp để đào tạo các mô hình ngôn ngữ tổng quát hoặc sử dụng trong các ứng dụng thực tế.

    Ví dụ về câu "anh ta đã uống một cốc nước" có thể có trong Wikipedia, nhưng không phải là một phần quan trọng của ngôn ngữ thông thường của người Việt Nam. Trong thực tế, khi giao tiếp hàng ngày, người Việt Nam có thể sử dụng các cụm từ khác để miêu tả hành động uống nước, ví dụ như "anh ta đã uống ly nước", "anh ta đã uống một ngụm nước" hoặc "anh ta đã uống một cốc nước lớn". Do đó, nếu chỉ sử dụng dữ liệu từ Wikipedia để huấn luyện mô hình ngôn ngữ, mô hình sẽ không hiểu được các biến thể khác nhau của cùng một hành động trong ngôn ngữ thông thường. [↑](#footnote-ref-11)
11. Syllable: âm tiết (đơn vị phát âm tối thiểu của một từ). Word token là từ.

    Ví dụ: Nghiên, cứu là âm tiết, nghiên cứu là một từ. [↑](#footnote-ref-12)
12. Các âm tiết tạo thành một từ trong tiếng việt dùng khoảng trắng để phân biệt, từ cũng dùng khoảng trắng để phân biệt giữa các từ (85% từ tiếng Việt bao gồm ít nhất 2 âm tiết trở nên), điều này không có trong tiếng Anh. [↑](#footnote-ref-13)
13. Phương pháp mã hóa Byte-Pair (BPE) là một phương pháp nén mã hóa và mở rộng từ vựng được sử dụng trong các mô hình ngôn ngữ dựa trên kết quả tiền huấn luyện. Nó hoạt động bằng cách kết hợp các cặp từ liên tiếp nhau xuất hiện nhiều nhất trong tập dữ liệu, sau đó ghép các cặp từ này lại thành một từ mới và lặp lại quá trình cho đến khi đạt được một kích thước từ vựng mong muốn hoặc khi không còn cặp từ nào xuất hiện nhiều nhất. Các từ mới này sẽ được sử dụng để biểu diễn các văn bản và từ mới nào không xuất hiện trong từ vựng sẽ được mã hóa bằng các ký hiệu đặc biệt. Phương pháp BPE cho phép mô hình xử lý được các từ mới và hiệu quả hơn trong việc nén dữ liệu so với các phương pháp mã hóa từ vựng truyền thống. Chi tiết tại mục 2-Introduction tài liệu Related Knowledge [↑](#footnote-ref-14)
14. fairseq là một framework được phát triển bởi Facebook AI Research (FAIR) cho các ứng dụng dịch máy, nhận diện giọng nói và xử lý ngôn ngữ tự nhiên, chuyên hỗ trợ các nghiên cứu và dự án liên quan đến model seq2seq. [↑](#footnote-ref-15)
15. transformers là một framework, được phát triển bởi Hugging Face. Transformers cung cấp một tập hợp các công cụ cho việc xử lý văn bản và xây dựng các mô hình cho các tác vụ như dịch máy, phân loại văn bản, sinh văn bản, và phân tích cảm xúc. [↑](#footnote-ref-16)
16. RoBERTa là một mô hình ngôn ngữ được phát triển bởi Facebook AI Research (FAIR). Nó là một phiên bản cải tiến của mô hình BERT và được đào tạo trên một lượng lớn dữ liệu hơn so với BERT. RoBERTa là một mô hình đưa ra các cải tiến đáng kể so với BERT. RoBERTa sử dụng một loạt các kỹ thuật tinh chỉnh để cải thiện quá trình huấn luyện và nâng cao hiệu suất của mô hình, bao gồm điều chỉnh các siêu tham số, tăng kích thước mô hình, mở rộng tập dữ liệu huấn luyện và thực hiện các quá trình tiền huấn khác. Chi tiết tại mục 1- PhoBERT tài liệu Related Knowledge. [↑](#footnote-ref-17)
17. RDRSegmenter là một công cụ phân đoạn từ và câu cho tiếng Việt, sử dụng phương pháp RDR (Recursion Dependency Reduction) để phân đoạn câu và từ trong tiếng Việt. [↑](#footnote-ref-18)
18. VnCoreNLP là một bộ công cụ phân tích ngôn ngữ tự nhiên cho tiếng Việt, bao gồm các tính năng như phân tích cú pháp, phân đoạn từ và câu, nhận dạng thực thể, tách từ đồng âm. [↑](#footnote-ref-19)
19. fastBPE là một công cụ được sử dụng để tạo ra các đơn vị phân tách dựa trên các subword. Nó là một phương pháp tạo subword nhanh chóng và hiệu quả, phụ thuộc vào việc xây dựng một bộ từ điển tần suất cực lớn. fastBPE đã được sử dụng để phân tách các câu đã được phân đoạn từ và câu bởi RDRSegmenter thành các subword units, từ đó tạo ra dữ liệu đầu vào cho việc huấn luyện mô hình RoBERTa cho tiếng Việt. Là package hỗ trợ tokenize từ (word) thành các từ phụ (subwords). [↑](#footnote-ref-20)
20. Subword units là các đơn vị cấu tạo thành từ tiền tố, hậu tố và các ký tự tạo thành từ có thể xuất hiện ở vị trí giữa các từ. Chúng được sử dụng để biểu diễn từ trong mô hình ngôn ngữ máy học và giúp cho mô hình có thể xử lý được các từ mới chưa xuất hiện trong từ điển. Ví dụ: Từ "unbelievable" có thể được tách thành "un", "believ", và "able". Sau khi tách thành các subword units, các từ có thể được biểu diễn bằng các vector trong không gian embedding, giúp cho mô hình hiểu được cấu trúc ngữ pháp và ngữ nghĩa của từ. [↑](#footnote-ref-21)
21. Tối ưu hóa (Optimization): Quá trình điều chỉnh các tham số của mô hình sao cho mô hình đạt được hiệu suất tốt nhất trên tập dữ liệu huấn luyện. Trong quá trình huấn luyện, các thông số của mô hình được điều chỉnh sao cho mô hình đạt được mức độ chính xác cao nhất trên tập dữ liệu huấn luyện. Điều này đảm bảo rằng mô hình sẽ hoạt động tốt hơn khi được sử dụng để dự đoán trên các tập dữ liệu mới. [↑](#footnote-ref-22)
22. Adam là một thuật toán tối ưu hóa gradient descent được sử dụng rộng rãi trong deep learning. Nó là một thuật toán tối ưu hóa tự điều chỉnh, tức là nó tự động thay đổi learning rate (tốc độ học) cho mỗi tham số của mô hình, dựa trên thông tin gradient của chúng. Chi tiết hơn tại mục 2- PhoBERT tài liệu Related Knowledge. [↑](#footnote-ref-23)
23. Epoch là một khái niệm trong machine learning, deep learning... Nó là một vòng lặp huấn luyện toàn bộ tập dữ liệu cho một mô hình học máy cụ thể. Trong mỗi epoch, mô hình sẽ được huấn luyện với các dữ liệu huấn luyện khác nhau trong tập dữ liệu đó. Sau mỗi epoch, các tham số của mô hình được cập nhật để cải thiện độ chính xác và giảm thiểu sai số trên tập dữ liệu huấn luyện. Thông thường, số lượng epoch được thiết lập trước và có thể được điều chỉnh để cải thiện độ chính xác của mô hình. [↑](#footnote-ref-24)
24. Nội dung tối ưu hóa bao gồm: Các bước tối ưu, lí do của từng bước được trình bày tại mục 3 – PhoBERT tài liệu Related Knowledge [↑](#footnote-ref-25)
25. "Downstream task datasets" là các tập dữ liệu được sử dụng để đánh giá khả năng ứng dụng của mô hình ngôn ngữ đã được huấn luyện trước đó. Trong quá trình huấn luyện mô hình, ta sử dụng một tập dữ liệu lớn để "huấn luyện" cho mô hình, sau đó ta sử dụng các tập dữ liệu khác (gọi là "downstream task datasets") để kiểm tra khả năng áp dụng của mô hình trên các tác vụ cụ thể như phân loại văn bản, dịch máy, phân tích cú pháp, và cung cấp đầu ra cho các ứng dụng thực tế.

    Để đánh giá đầy đủ khả năng của một pre-trained model, nó cần phải được đánh giá trên các nhiệm vụ cụ thể (có thể được gọi là "downstream tasks"), chẳng hạn như phân loại văn bản hoặc nhận dạng thực thể. Các nhiệm vụ này thường có dữ liệu huấn luyện nhỏ hơn và được giải quyết trực tiếp với mô hình pre-trained. Việc đánh giá trên các downstream tasks giúp đánh giá chính xác hơn hiệu suất của mô hình pre-trained và đưa ra một cái nhìn tổng quan về khả năng áp dụng thực tế của nó. [↑](#footnote-ref-26)
26. VLSP 2013 là bộ dữ liệu tiếng Việt cho tác vụ gán nhãn từ loại và gán nhãn ngữ nghĩa. [↑](#footnote-ref-27)
27. VnDT v1.1 là một tập dữ liệu phụ thuộc cú pháp tiếng Việt. Bộ dữ liệu này cung cấp cấu trúc phụ thuộc cây cho một câu bằng cách xác định các từ phụ thuộc (dependencies) của mỗi từ trong câu. [↑](#footnote-ref-28)
28. VLSP 2016 là bộ dữ liệu tiếng Việt cho tác vụ nhận dạng thực thể được sử dụng làm benchmark cho tác vụ nhận dạng thực thể tiếng Việt. [↑](#footnote-ref-29)
29. NLI (XNLI) corpus v1.0 là một tập dữ liệu được thực hiện để đánh giá khả năng phân loại câu tương đồng qua các ngôn ngữ khác nhau. Tập dữ liệu này được sử dụng trong các nghiên cứu liên ngôn ngữ để đánh giá khả năng của các mô hình ngôn ngữ trước khi chúng được fine-tuning cho các tác vụ cụ thể trong một ngôn ngữ cụ thể. Tập dữ liệu này bao gồm các cặp câu trong nhiều ngôn ngữ, bao gồm cả tiếng Anh và tiếng Việt. [↑](#footnote-ref-30)
30. Fine-tuning là quá trình tiếp tục huấn luyện một mô hình học sâu đã được huấn luyện sẵn trên một tập dữ liệu nhất định để đạt được hiệu suất tốt hơn trên tập dữ liệu mới cho một tác vụ nhất định. Fine-tuning thực hiện điều chỉnh một số thông số của mô hình để phù hợp với tập dữ liệu mới cho tác vụ tương ứng. [↑](#footnote-ref-31)
31. Trong các thử nghiệm sơ bộ của chúng tôi, việc sử dụng trung bình các embedding có bối cảnh của các subword token để đại diện cho từ gây ra hiệu suất thấp hơn một chút so với việc sử dụng embedding có bối cảnh của subword đầu tiên. [↑](#footnote-ref-32)
32. Biaffine dependency parser là một kiểu parser cú pháp (syntax parser) dựa trên mô hình neural network. Biaffine dependency parser sử dụng một kiến trúc mạng neural truyền thẳng (feedforward neural network) để tính toán xác suất của các cặp từ liên kết trong câu, từ đó xác định các mối quan hệ phụ thuộc giữa các từ trong câu. [↑](#footnote-ref-33)
33. "embedding" là một kỹ thuật biểu diễn từ hoặc câu dưới dạng một vectơ số. Nhằm mục đích biến các từ hoặc câu (văn bản) thành các vectơ số để có thể sử dụng chúng trong các thuật toán máy học và xử lý ngôn ngữ tự nhiên khác. Các word embedding được tạo ra bằng cách sử dụng một mô hình máy học để học cách biểu diễn các từ dưới dạng các vectơ số, sao cho các từ có ý nghĩa tương đồng về mặt ngữ nghĩa sẽ có các vectơ số tương tự nhau. Các word embedding thường được sử dụng trong các nhiệm vụ như phân loại văn bản, tìm kiếm thông tin và dịch thuật. [↑](#footnote-ref-34)
34. Contextualized embedding là một loại embedding trong xử lý ngôn ngữ tự nhiên được học bằng mô hình ngôn ngữ biểu diễn của Transformer, cho phép biểu diễn từ dựa trên bối cảnh xung quanh nó trong câu. Nó cho phép mô hình học được các đặc trưng ngữ nghĩa của từ, phù hợp với ngữ cảnh và ngữ nghĩa của câu. [↑](#footnote-ref-35)
35. Transformers là một kiến trúc mạng nơ-ron mạnh mẽ. Nó sử dụng một kiến trúc self-attention để xử lý các chuỗi đầu vào và học cách giải quyết các tác vụ xử lý ngôn ngữ tự nhiên. Chi tiết tại mục 1 – PhoBERT tài liệu Related Knowledge. [↑](#footnote-ref-36)
36. AdamW là một phương pháp tối ưu hóa gradient dựa trên Adam được sử dụng để cập nhật trọng số của mạng neural. Nó thêm một hạng mục trọng số L2 regularization vào thuật toán Adam để giúp mô hình học được một phân bố trọng số tốt hơn và giảm thiểu hiện tượng overfitting trong quá trình huấn luyện. [↑](#footnote-ref-37)
37. jointWPD: Là một mô hình neural cho việc gán nhãn POS và phân tích cú pháp dependency parsing cho tiếng Việt. [↑](#footnote-ref-38)
38. VnCoreNLP-POS: Là một mô hình dựa trên đặc trưng cho việc gán nhãn POS trong tiếng Việt [↑](#footnote-ref-39)
39. LAS: Là viết tắt của Labeled Attachment Score, đây là một độ đo cho chất lượng của phân tích cú pháp syntactic parsing, tính toán tỉ lệ số phân tích đúng mà mô hình đã đưa ra và số cây cú pháp đúng trong tập dữ liệu thử nghiệm. [↑](#footnote-ref-40)
40. UAS: Là viết tắt của Unlabeled Attachment Score, cũng là một độ đo cho chất lượng của phân tích cú pháp syntactic parsing, nhưng chỉ tính toán tỉ lệ số phân tích đúng mà mô hình đã đưa ra và số từ kết nối đúng giữa các nút trong cây cú pháp trong tập dữ liệu thử nghiệm. [↑](#footnote-ref-41)
41. BiLSTM-CNN-CRF là một mô hình Deep Learning được sử dụng trong xử lý ngôn ngữ tự nhiên, kết hợp giữa mạng neuron LSTM và mạng CNN để trích xuất đặc trưng và một mô hình CRF để thực hiện gán nhãn. [↑](#footnote-ref-42)
42. ETNLP: Là viết tắt của Ensemble Transfer NLP, đây là một phương pháp để cải thiện hiệu suất của mô hình NLP bằng cách sử dụng các mô hình khác nhau và kết hợp kết quả của chúng. [↑](#footnote-ref-43)
43. XLMMLM+TLM là một kiến trúc mô hình mạng học sâu dựa trên kiến trúc BERT. Được đào tạo trên tập dữ liệu lớn với các ngôn ngữ khác nhau, nhằm giải quyết vấn đề đa ngôn ngữ. Mô hình này sử dụng hai mục tiêu huấn luyện chính: mô hình ngôn ngữ đa ngôn ngữ (MLM) và mô hình dịch đa ngôn ngữ (TLM). [↑](#footnote-ref-44)
44. Để tinh chỉnh XLM-R, chúng tôi sử dụng phương pháp grid search trên tập validation để lựa chọn tốc độ học AdamW từ {5e-6, 1e-5, 2e-5, 4e-5} và kích thước batch từ {16, 32}

    Grid search là một kỹ thuật tìm kiếm siêu tham số để tìm các giá trị tối ưu cho các siêu tham số của mô hình. Grid search được sử dụng để tự động hóa việc tìm kiếm các giá trị siêu tham số tốt nhất bằng cách xác định một tập hợp các giá trị để thử nghiệm và kiểm tra hiệu suất của mô hình trên từng giá trị siêu tham số trong lưới. Kết quả sẽ cho ta biết giá trị tối ưu của từng siêu tham số để cải thiện hiệu suất của mô hình. [↑](#footnote-ref-45)