TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN CUỐI KÌ**

**NHẬP MÔN**

**XỬ LÝ NGÔN NGỮ TỰ NHIÊN**

*Người hướng dẫn*: **PGS.TS. LÊ ANH CƯỜNG**

*Người thực hiện*: **NGUYỄN THỊ CẨM THÙY – 52100844**

**ĐINH PHƯƠNG MY – 52100703**

**NGUYỄN PHÚC TRỌNG – 52100857**

*Lớp:* **21050201**

**21050401**

*Khoá:* **25**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN CUỐI KÌ**

**NHẬP MÔN**

**XỬ LÝ NGÔN NGỮ TỰ NHIÊN**

*Người hướng dẫn*: **PGS.TS. LÊ ANH CƯỜNG**

*Người thực hiện*: **NGUYỄN THỊ CẨM THÙY – 52100844**

**ĐINH PHƯƠNG MY – 52100703**

**NGUYỄN PHÚC TRỌNG – 52100857**

*Lớp:* **21050201**

**21050401**

*Khoá:* **25**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024**

**LỜI CẢM ƠN**

Lời đầu tiên, nhóm 10 xin gửi lời cảm ơn đến khoa Công nghệ thông tin đã tạo ra môn học này cho nhóm 10 có thể học tập cùng nhau và thực hành môn xử lý ngôn ngữ tự nhiên. Nhóm 10 xin chân thành cảm ơn PSG.TS.Lê Anh Cường đã giảng dạy, truyền đạt lại kiến thức một cách rất nhiệt tình và tạo ra bầu không khí học tập thoải mái cho nhóm 10 học tập tốt.

Nhóm 10 biết bài báo cáo này còn nhiều điều còn thiếu sót, nhóm 10 mong thầy có thể góp ý cho nhóm em.

Nhóm 10 xin chân thành cảm ơn!

**ĐỒ ÁN ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Nhóm 10 xin cam đoan đây là sản phẩm đồ án của riêng nhóm và được sự hướng dẫn của PGS.TS.Lê Anh Cường. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào nhóm xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do nhóm gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày 25 tháng 03 năm 2024*

*Tác giả*

*(ký tên và ghi rõ họ tên)*



*Nguyễn Thị Cẩm Thùy*

*Đinh Phương My*

*Nguyễn Phúc Trọng*

**PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN**

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(ký và ghi họ tên)

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(ký và ghi họ tên)

**TÓM TẮT**

Báo cáo này được chia thành 3 chương, mỗi chương tập trung vào một phần quan trọng của nghiên cứu. Dưới đây là một mô tả nhanh về từng chương:

Chương 1. Giới thiệu và trình bày về các biểu diễn token theo phương pháp BPE, cùng với ví dụ về các mô hình sử dụng tokenizer theo BPE. Sau đó, so sánh hai mô hình trên một bài toán cụ thể có sử dụng BPE và không sử dụng BPE, với điều kiện cả hai mô hình được huấn luyện và kiểm tra trên cùng một bộ dữ liệu.

Chương 2. Tập trung vào việc xây dựng một bài toán phát hiện và sửa lỗi chính tả tiếng Việt thông qua học sâu. Các nhiệm vụ bao gồm: chọn mô hình (mô hình chỉ có Encoder, mô hình sinh Encoder-Decoder, hoặc mô hình chỉ sử dụng Decoder), xây dựng và sinh dữ liệu huấn luyện từ các loại lỗi chính tả tiếng Việt phổ biến, huấn luyện mô hình (có thể sử dụng Pre-trained model và thực hiện fine-tune hoặc tự train từ đầu), và đánh giá độ chính xác của mô hình.

Chương 3. Đánh giá khả năng thực hiện của mô hình, bao gồm cả những thách thức và giới hạn của quá trình nghiên cứu. Chương này là nơi tóm tắt kết quả, đưa ra đề xuất cho các nghiên cứu tiếp theo và thảo luận về những khó khăn gặp phải trong quá trình nghiên cứu.

**MỤC LỤC**

[LỜI CẢM ƠN i](#_heading=h.30j0zll)

[PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN iii](#_heading=h.1fob9te)

[TÓM TẮT iv](#_heading=h.3znysh7)

[DANH MỤC CÁC CHỮ VIẾT TẮT 3](#_heading=h.tyjcwt)

[DANH CHỨC NĂNG CÁC BẢNG BIỂU, HÌNH ẢNH 4](#_heading=h.3dy6vkm)

[CHƯƠNG 1 – PHƯƠNG PHÁP BPE 6](#_heading=h.1t3h5sf)

[1.1 Sơ lược về BPE 6](#_heading=h.4d34og8)

[1.2 Các bước hoạt động 7](#_heading=h.2s8eyo1)

[1.3 Mô hình ứng dụng 7](#_heading=h.17dp8vu)

[1.4 Ứng dụng vào xây dựng bài toán 8](#_heading=h.26in1rg)

[1.4.1 Mô tả bài toán 8](#_heading=h.lnxbz9)

[1.4.2. Tiền xử lí dữ liệu 8](#_heading=h.35nkun2)

[1.4.3 Xây dựng mô hình 10](#_heading=h.1y810tw)

[1.4.3.1 Mô hình 1: Không sử dụng BPE 10](#_heading=h.4i7ojhp)

[1.4.3.2 Mô hình 2: Sử dụng BPE 12](#_heading=h.49x2ik5)

[1.4.4 Phân loại nhãn cho một review mới 15](#_heading=h.2grqrue)

[CHƯƠNG 2 – BÀI TOÁN PHÁT HIỆN VÀ SỬA CHỮA LỖI CHÍNH TẢ TIẾNG VIỆT 18](#_heading=h.2u6wntf)

[2.1 Sơ lược về Seq2seq 18](#_heading=h.19c6y18)

[2.2 Các hoạt động trong mã hóa và giải mã 18](#_heading=h.3tbugp1)

[2.3 Mô hình ứng dụng 19](#_heading=h.28h4qwu)

[2.4 Ứng dụng vào xây dựng bài toán 19](#_heading=h.nmf14n)

[2.4.1 Mô tả bài toán 19](#_heading=h.37m2jsg)

[2.4.2. Xây dựng bài toán 19](#_heading=h.1mrcu09)

[2.4.2.1 Chuẩn bị data 19](#_heading=h.46r0co2)

[2.4.2.2 Chuẩn bị danh sách các lỗi thường gặp 21](#_heading=h.111kx3o)

[2.4.2.3 Hàm phát sinh lỗi chính tả thường gặp 22](#_heading=h.2zbgiuw)

[2.4.2.4 Tách câu thành các cụm từ 25](#_heading=h.3cqmetx)

[2.4.2.5 Tách các cụm từ thành danh sách 5-grams với MAXLEN =39 27](#_heading=h.4bvk7pj)

[2.4.2.6 Hàm encoder và decoder 28](#_heading=h.3q5sasy)

[2.4.2.7 Xây dựng mô hình 29](#_heading=h.kgcv8k)

[2.4.2.8 Tách dữ liệu thành tập training và validation 31](#_heading=h.2iq8gzs)

[2.4.2.9 Huấn luyện mô hình 33](#_heading=h.4h042r0)

[2.4.2.10 Ứng dụng mô hình 34](#_heading=h.1baon6m)

[CHƯƠNG 3 – KẾT LUẬN 39](#_heading=h.2nusc19)

[3.1 Kết quả đạt được 39](#_heading=h.1302m92)

[3.2 Thuận lời và khó khăn 39](#_heading=h.3mzq4wv)

[3.2.1 Thuận lợi 39](#_heading=h.2250f4o)

[3.2.2 Khó khăn 39](#_heading=h.319y80a)

[3.3 Đánh giá 39](#_heading=h.1gf8i83)

[TÀI LIỆU THAM KHẢO 41](#_heading=h.upglbi)

**DANH MỤC CÁC CHỮ VIẾT TẮT**

**CÁC CHỮ VIẾT TẮT**

API Application Programming Interface

BPE Byte Pair Encoding

BERT Bidirectional Encoder Representations from Transformers

IMDb Internet Movie Database

GPT-2 Generative Pre-trained Transformer Version 2

NLP Natural Language Processing

SVM Support Vector Machine

**DANH CHỨC NĂNG CÁC BẢNG BIỂU, HÌNH ẢNH**

**DANH CHỨC NĂNG BẢNG**

[Bảng 4.3 Đánh giá độ hoàn thành đồ án 19](#_heading=h.2fk6b3p)

**DANH CHỨC NĂNG HÌNH ẢNH**

[Hình 1.4.1 Tải tập dữ liệu IMDb của 50k bài đánh giá phim 8](#_heading=h.1ksv4uv)

[Hình 1.4.2 Dữ liệu IMDb của 50k bài đánh giá phim 8](#_heading=h.44sinio)

[Hình 1.4.3 Định nghĩa hàm preprocess\_text 9](#_heading=h.2jxsxqh)

[Hình 1.4.4 Áp dụng hàm preprocess\_text 9](#_heading=h.z337ya)

[Hình 1.4.5 Dữ liệu đã được tiền xử lý 10](#_heading=h.3j2qqm3)

[Hình 1.4.6 Tokenized dữ liệu cho mô hình Word2Vec 10](#_heading=h.2xcytpi)

[Hình 1.4.7 Xây dựng hàm get\_vector 11](#_heading=h.3whwml4)

[Hình 1.4.8 Dữ liệu đã thêm cột tokenized và word2vec 11](#_heading=h.2bn6wsx)

[Hình 1.4.9 Chuẩn bị dữ liệu và mô hình phân loại SVM được chọn 11](#_heading=h.qsh70q)

[Hình 1.4.10 Đánh giá mô hình SVM 12](#_heading=h.3as4poj)

[Hình 1.4.11 Kết quả đánh giá mô hình SVM 12](#_heading=h.1pxezwc)

[Hình 1.4.12 Tokenized dữ liệu cho mô hình BERT 12](#_heading=h.2p2csry)

[Hình 1.4.13 xây dựng hàm tokenize\_data 13](#_heading=h.147n2zr)

[Hình 1.4.14 Mã hóa và ánh xạ nhãn từ văn bản sang số 13](#_heading=h.3o7alnk)

[Hình 1.4.15 Chuẩn bị dữ liệu cho mô hình BERT 13](#_heading=h.23ckvvd)

[Hình 1.4.16 T ạo DataLoader cho mô hình BERT 14](#_heading=h.ihv636)

[Hình 1.4.17 Huấn luyện mô hình BERT 14](#_heading=h.32hioqz)

[Hình 1.4.18 Đánh giá mô hình BERT 15](#_heading=h.1hmsyys)

[Hình 1.4.19 Kết quả đánh giá mô hình BERT 15](#_heading=h.41mghml)

[Hình 1.4.20 Xây dựng hàm preprocess\_word2vec 16](#_heading=h.vx1227)

[Hình 1.4.21 Kết quả in nhãn của mô hình Word2Vec 16](#_heading=h.3fwokq0)

[Hình 1.4.22 Xây dựng hàm preprocess\_bert 16](#_heading=h.1v1yuxt)

[Hình 1.4.23 Kết quả in nhãn của mô hình BERT 16](#_heading=h.4f1mdlm)

[Hình ảnh 2.4.1 Chuẩn bị data 20](#_heading=h.2lwamvv)

[Hình ảnh 2.4.2 Danh sách lỗi đánh máy 21](#_heading=h.3l18frh)

[Hình ảnh 2.4.3 Danh sách lỗi vùng miền 22](#_heading=h.206ipza)

[Hình ảnh 2.4.4 Danh sách lỗi viết tắt 22](#_heading=h.4k668n3)

[Hình ảnh 2.4.5 Hàm \_teen\_code 22](#_heading=h.1egqt2p)

[Hình ảnh 2.4.6 Hàm \_add\_noise (p1) 23](#_heading=h.3ygebqi)

[Hình ảnh 2.4.7 Hàm \_add\_noise (p2) 24](#_heading=h.2dlolyb)

[Hình ảnh 2.4.8 Ký tự trong encoder 25](#_heading=h.sqyw64)

[Hình ảnh 2.4.9 Tách câu thành cụm từ 26](#_heading=h.1rvwp1q)

[Hình ảnh 2.4.10 Tách các cụm từ thành danh sách 5-grams với maxlen là 39. 27](#_heading=h.1664s55)

[Hình ảnh 2.4.11 Hàm \_encoder\_data và \_decoder\_data 28](#_heading=h.25b2l0r)

[Hình ảnh 2.4.11 Import thư viện cần thiết cho xây dựng model 29](#_heading=h.34g0dwd)

[Hình ảnh 2.4.12 Xây dựng encoder và decoder 30](#_heading=h.1jlao46)

[Hình ảnh 2.4.13 Thêm các layer vào model 31](#_heading=h.43ky6rz)

[Hình ảnh 2.4.14 Chia dữ liệu 31](#_heading=h.xvir7l)

[Hình ảnh 2.4.15 Hàm \_generate\_data 32](#_heading=h.3hv69ve)

[Hình ảnh 2.4.16 Tạo train\_generator và validation\_generator 33](#_heading=h.1x0gk37)

[Hình ảnh 2.4.17 Huấn luyện model 33](#_heading=h.2w5ecyt)

[Hình ảnh 2.4.18 Import thư viện cần thiết và load model 34](#_heading=h.3vac5uf)

[Hình ảnh 2.4.19 Hàm nltk\_ngrams 34](#_heading=h.2afmg28)

[Hình ảnh 2.4.20 Hàm guess 35](#_heading=h.pkwqa1)

[Hình ảnh 2.4.21 Hàm \_add\_punctuation 36](#_heading=h.39kk8xu)

[Hình ảnh 2.4.22 Hàm \_correct 37](#_heading=h.1opuj5n)

[Hình ảnh 2.4.23 Test model 38](#_heading=h.48pi1tg)

**CHƯƠNG 1 – PHƯƠNG PHÁP BPE**

**1.1 Sơ lược về BPE**

1.1.1 Định nghĩa

Byte-Pair Encoding (BPE) là một thuật toán nén được sử dụng trong Xử lý ngôn ngữ tự nhiên (NLP) để thể hiện vốn từ vựng lớn với một tập hợp nhỏ các đơn vị từ con (subword). Nó được giới thiệu bởi Sennrich vào năm 2016 và đã được sử dụng rộng rãi trong các tác vụ NLP khác nhau như dịch máy, phân loại văn bản và tạo văn bản.

Ý tưởng cơ bản của BPE là hợp nhất lặp đi lặp lại cặp byte hoặc ký tự liên tiếp thường xuyên nhất trong một kho văn bản cho đến khi đạt được kích thước từ vựng được xác định trước. Các đơn vị con từ kết quả có thể được sử dụng để đại diện cho văn bản gốc một cách nhỏ gọn và hiệu quả hơn.

1.1.2 Thành phần

* Vocabulary: Một tập hợp các đơn vị subword có thể được sử dụng để đại diện cho một kho văn bản.
* Byte: Một đơn vị thông tin kỹ thuật số thường bao gồm tám bit.
* Character: Một ký hiệu đại diện cho một chữ cái hoặc chữ số được viết hoặc in.
* Frequency: Số lần một byte hoặc ký tự xảy ra trong một kho văn bản.
* Merge: Quá trình kết hợp hai byte hoặc ký tự liên tiếp để tạo ra một subword mới.

1.1.3 Ưu điểm so với các phương pháp biểu diễn token truyền thống

So với các phương pháp biểu diễn token truyền thống, BPE sở hữu nhiều ưu điểm nổi bật, đem lại nhiều lợi ích đáng chú ý như sau:

* Khả năng xử lý từ vựng mở: BPE có khả năng biểu diễn các từ mới một cách hiệu quả bằng cách kết hợp các token con đã có trong từ vựng. Điều này đồng nghĩa với việc không cần phải bổ sung thủ công từ vựng mới mỗi khi gặp từ mới trong dữ liệu.
* Giảm thiểu lỗi chính tả: Nhờ khả năng chia nhỏ các từ sai chính tả thành các token con chính xác, BPE góp phần nâng cao hiệu quả xử lý lỗi chính tả trong văn bản.
* Cải thiện hiệu suất mô hình: Sử dụng BPE có thể giảm thiểu kích thước của từ vựng, đồng thời cải thiện khả năng biểu diễn các từ mới. Kết quả là, mô hình NLP có thể hoạt động hiệu quả hơn và đạt được kết quả tốt hơn trong các tác vụ như phân loại văn bản, dự đoán từ tiếp theo, và dịch máy.

**1.2 Các bước hoạt động**

* Bước 1: Khởi tạo từ vựng với tất cả các byte hoặc ký tự trong kho văn bản
* Bước 2: Tính tần suất của từng byte hoặc ký tự trong kho văn bản.
* Bước 3: Lặp lại các bước sau cho đến khi đạt được kích thước từ vựng mong muốn:
* Tìm cặp byte hoặc ký tự liên tiếp thường xuyên nhất trong kho văn bản
* Hợp nhất cặp để tạo một đơn vị từ con mới.
* Cập nhật số tần suất của tất cả các byte hoặc ký tự có chứa cặp đã phối.
* Thêm đơn vị từ con mới vào từ vựng.
* Bước 4: Trình bày kho văn bản bằng cách sử dụng các đơn vị từ phụ trong từ vựng.

**1.3 Mô hình ứng dụng**

Nhờ những ưu điểm vượt trội, BPE đã trở thành một công cụ quan trọng trong việc phát triển các mô hình NLP hiện đại, tiêu biểu như:

* Transformer: Transformer là một mô hình kiến trúc mạng nơ-ron được sử dụng cho nhiều nhiệm vụ xử lý ngôn ngữ tự nhiên, bao gồm dịch máy, tóm tắt văn bản và trả lời câu hỏi. Transformer sử dụng tokenizer BPE để biểu diễn văn bản đầu vào và đầu ra.
* GPT-2: GPT-2 là một mô hình ngôn ngữ lớn được đào tạo trên một tập dữ liệu khổng lồ gồm văn bản và mã. GPT-2 sử dụng tokenizer BPE để biểu diễn văn bản đầu vào và tạo văn bản đầu ra.
* BERT: BERT là một mô hình ngôn ngữ được đào tạo trước có thể được sử dụng cho nhiều nhiệm vụ xử lý ngôn ngữ tự nhiên, bao gồm phân loại văn bản, trích xuất thực thể được đặt tên và trả lời câu hỏi. BERT sử dụng tokenizer BPE để biểu diễn văn bản đầu vào.

**1.4 Ứng dụng vào xây dựng bài toán**

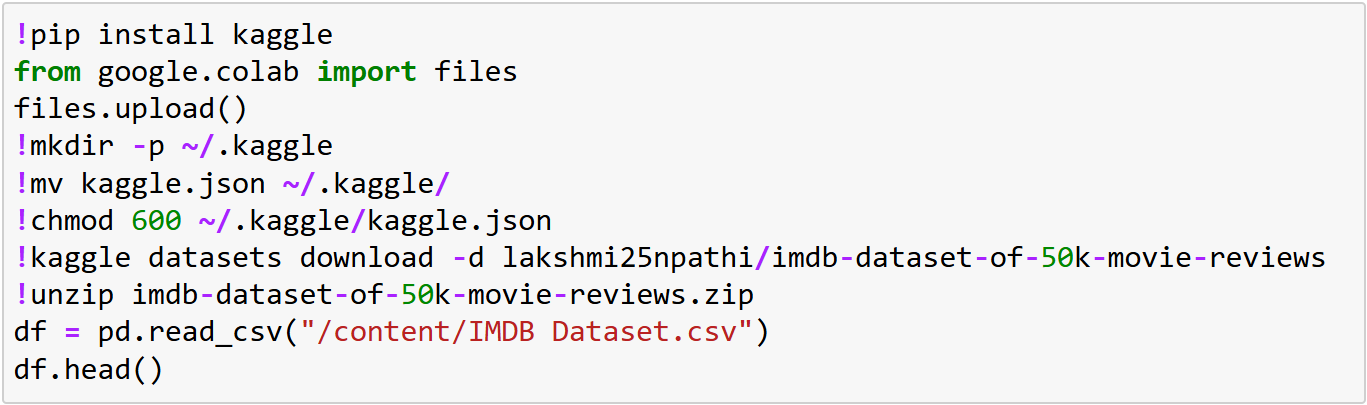
***1.4.1 Mô tả bài toán***

Bài toán sử dụng để so sánh hiệu suất của hai mô hình NLP là Phân loại đánh giá (Sentiment Analysis) trên tập dữ liệu “Sentiment Analysis from IMDB Reviews” được cung cấp bởi Kaggle.

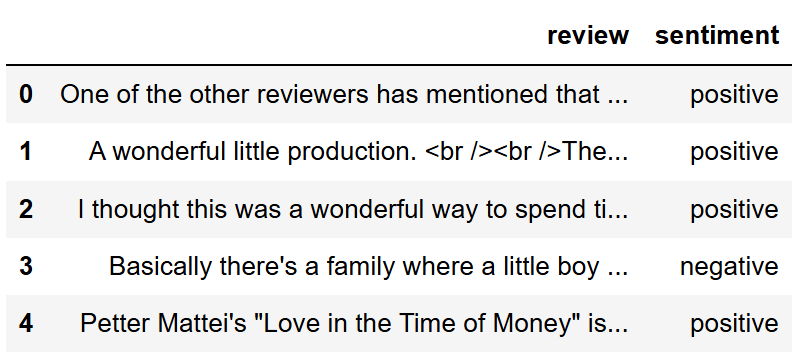
Mục tiêu của bài toán là phân loại đánh giá phim trong IMDB thành hai loại: tích cực (pos) và tiêu cực (neg). Đánh giá hiệu suất dựa trên các chỉ số như độ chính xác, precision, recall và F1-score giúp xác định mô hình nào hoạt động tốt hơn cho bài toán.

***1.4.2. Tiền xử lí dữ liệu***

Đầu tiên, cần tải tập dữ liệu từ Kaggle. Sử dụng Kaggle API để tải xuống tập dữ liệu IMDb của 50k bài đánh giá phim và sau đó giải nén nó.



Hình 1.4.1 Tải tập dữ liệu IMDb của 50k bài đánh giá phim

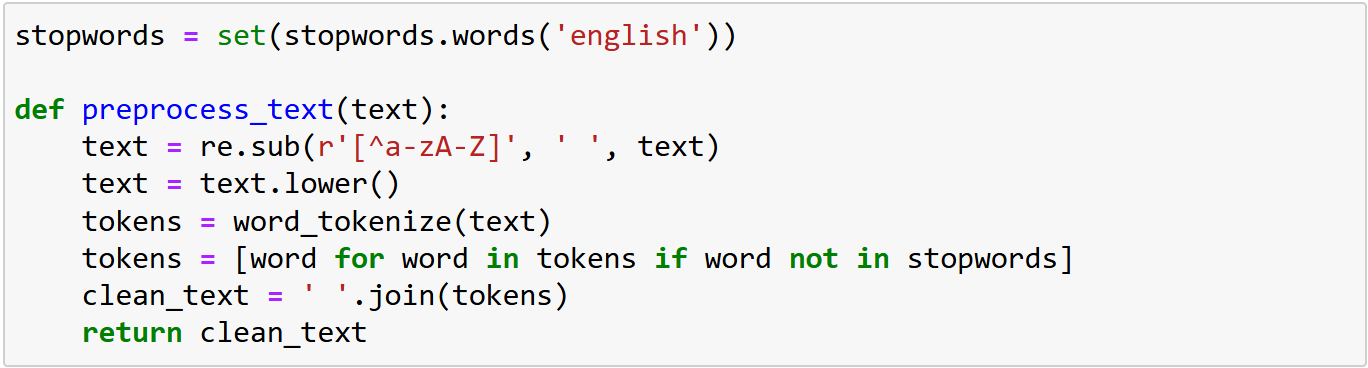


Hình 1.4.2 Dữ liệu IMDb của 50k bài đánh giá phim

Để tiến hành tiền xử lý văn bản trong cột “review” của DataFrame, chúng ta sẽ sử dụng thư viện NLTK để tải và loại bỏ stopwords của tiếng Anh. Stopwords là những từ phổ biến trong ngôn ngữ mà thường không mang ý nghĩa đặc biệt trong việc phân tích văn bản, chẳng hạn như “the”, “is”, “and”,...

Sau đó, định nghĩa hàm “preprocess\_text” để thực hiện các bước tiền xử lý như sau:

* Loại bỏ ký tự đặc biệt và số: Sử dụng biểu thức chính quy để thay thế tất cả các ký tự không phải là chữ cái bằng khoảng trắng.
* Chuyển đổi thành chữ thường: Sử dụng phương thức lower() để chuyển đổi tất cả các ký tự thành chữ thường.
* Tách từ: Sử dụng word\_tokenize từ thư viện NLTK để tách câu thành các từ riêng lẻ.
* Loại bỏ stopwords: Loại bỏ các từ stopwords khỏi danh sách các từ đã tách.
* Kết hợp các từ lại thành một câu: Sử dụng phương thức join() để kết hợp các từ lại thành một chuỗi.

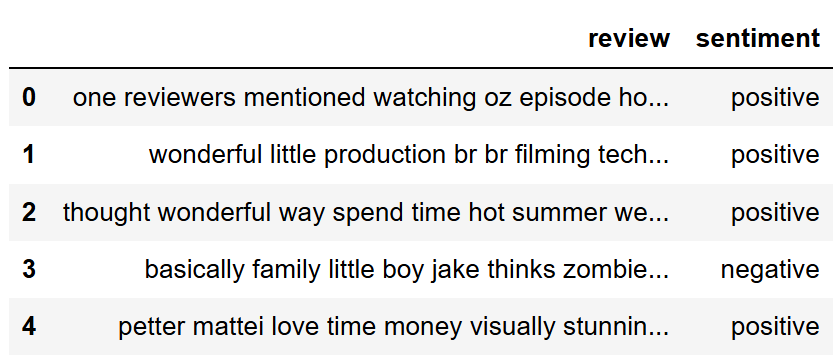


Hình 1.4.3 Định nghĩa hàm preprocess\_text

Cuối cùng, sử dụng phương thức apply() của DataFrame để áp dụng hàm preprocess\_text vào mỗi phần tử trong cột “review” của DataFrame data. Điều này giúp tiền xử lý văn bản trong cột “review” của dữ liệu.



Hình 1.4.4 Áp dụng hàm preprocess\_text



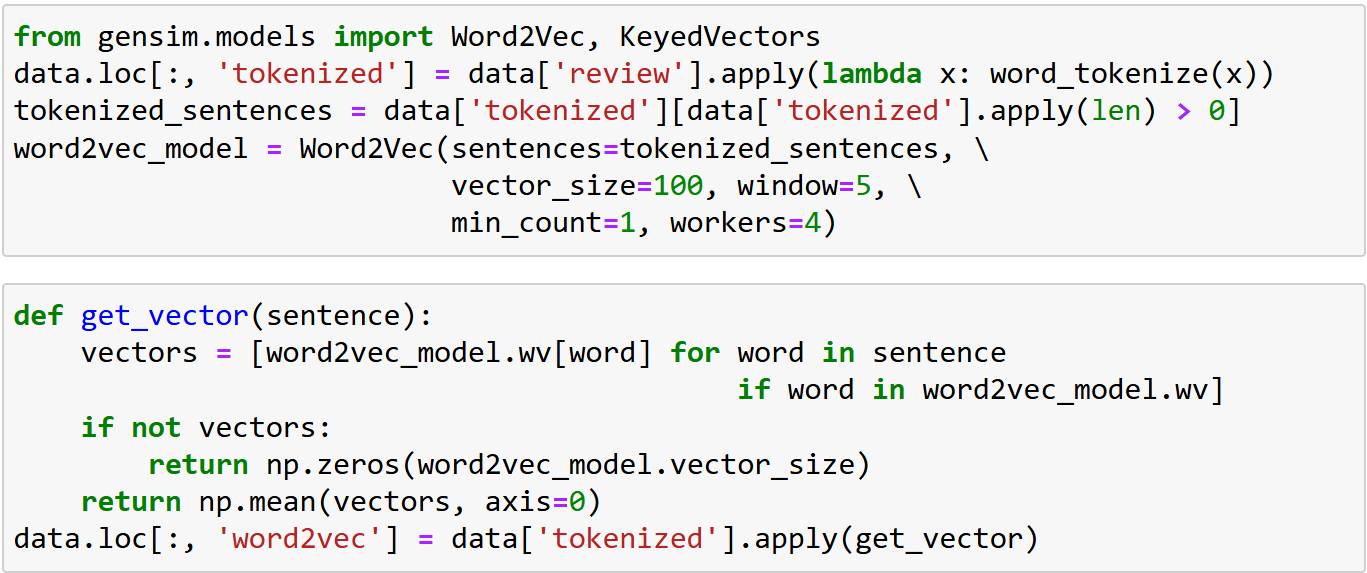
Hình 1.4.5 Dữ liệu đã được tiền xử lý

Kết quả của quá trình tiền xử lý là dữ liệu văn bản đã được làm sạch và chuẩn hóa, sẵn sàng cho các bước phân tích và xử lý tiếp theo. Dữ liệu đã qua tiền xử lý không chứa các ký tự đặc biệt và số, đã được chuyển thành chữ thường, từ đã được tách và loại bỏ các stopwords. Điều này giúp cải thiện chất lượng và hiệu suất của các mô hình hoặc phân tích văn bản tiếp theo.

***1.4.3 Xây dựng mô hình***

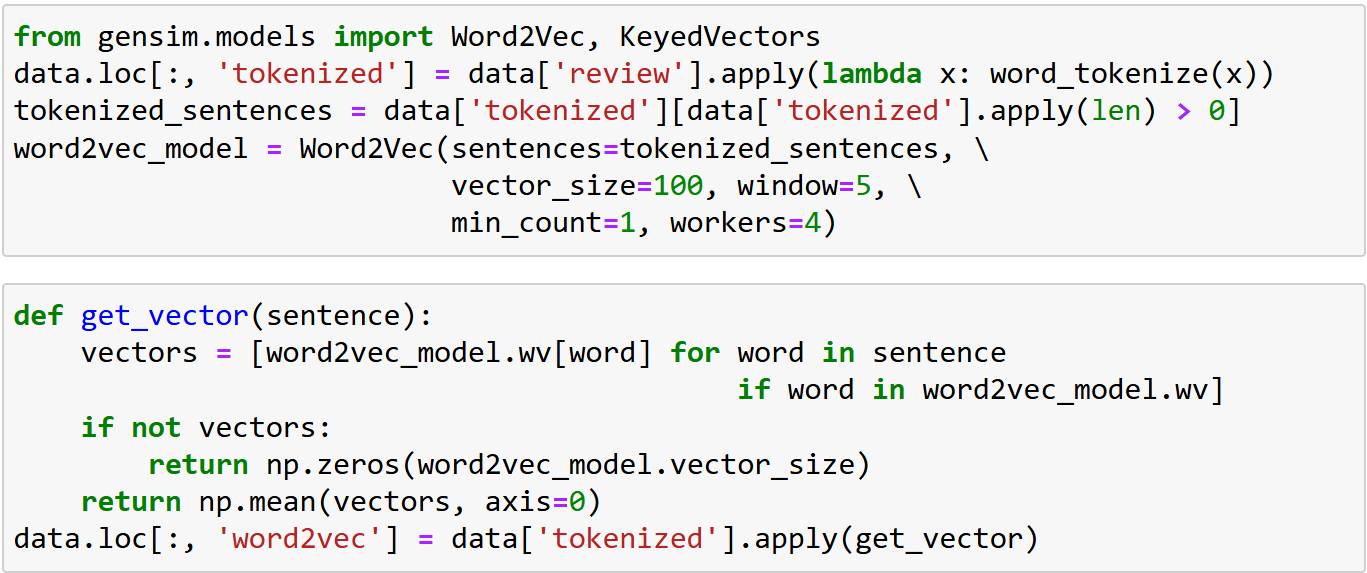
1.4.3.1 Mô hình 1: Không sử dụng BPE

Đầu tiên, dữ liệu văn bản trong cột “review” của DataFrame được tokenized bằng cách sử dụng word\_tokenize từ thư viện NLTK. Kết quả tokenized được lưu trong cột mới được gán cho data[‘tokenized’]. Sau đó, sử dụng thư viện Gensim để xây dựng một mô hình Word2Vec trên các câu đã tokenized từ dữ liệu.

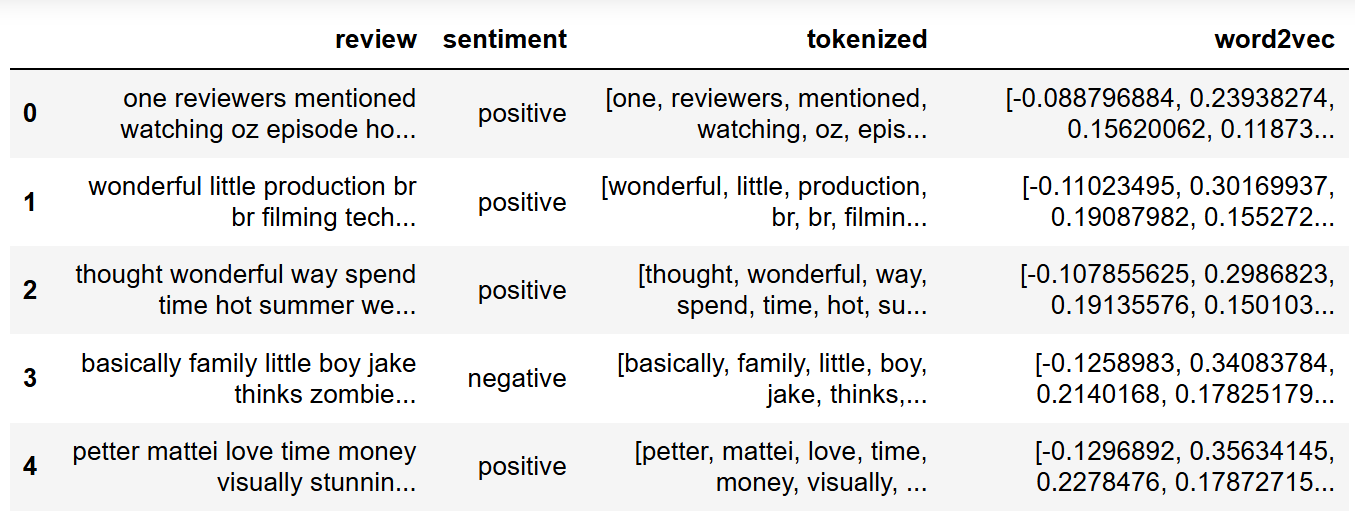


Hình 1.4.6 Tokenized dữ liệu cho mô hình Word2Vec

Tiếp theo, xây dựng hàm get\_vector được định nghĩa để trích xuất vector từ mô hình Word2Vec cho mỗi câu tokenized. Trong đó, mỗi từ trong câu được kiểm tra xem có trong từ điển của mô hình không, sau đó trích xuất vector của từ đó. Nếu không có từ nào trong từ điển của mô hình, trả về vector 0. Tất cả các vector của các từ trong câu được kết hợp lại và lấy trung bình để tạo ra một vector đại diện cho toàn bộ câu.

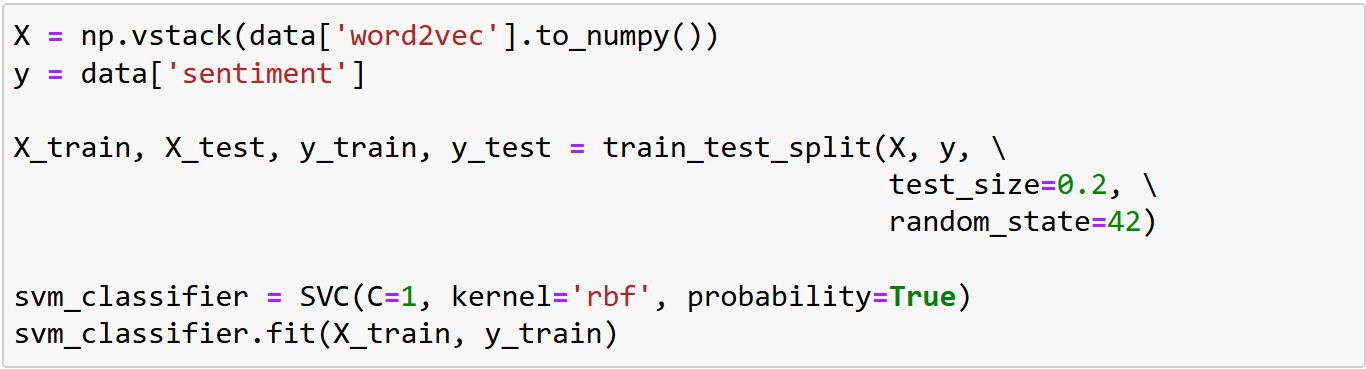


Hình 1.4.7 Xây dựng hàm get\_vector



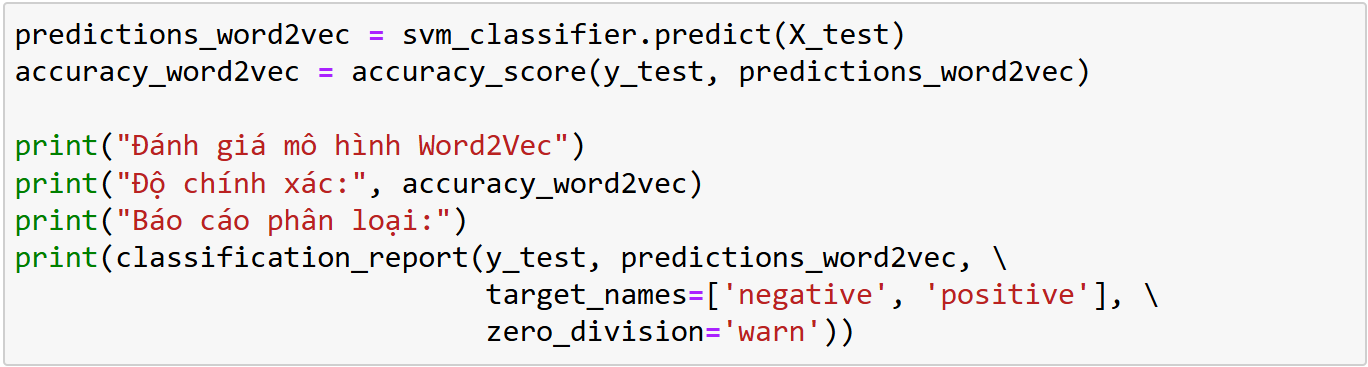
Hình 1.4.8 Dữ liệu đã thêm cột tokenized và word2vec

Chuẩn bị dữ liệu và huấn luyện mô hình với dữ liệu được chuẩn bị bằng cách chia thành tập huấn luyện và tập kiểm tra với tỷ lệ 80/20 sử dụng train\_test\_split từ thư viện sklearn. Một mô hình phân loại SVM từ thư viện sklearn với kernel RBF được chọn để huấn luyện trên dữ liệu.

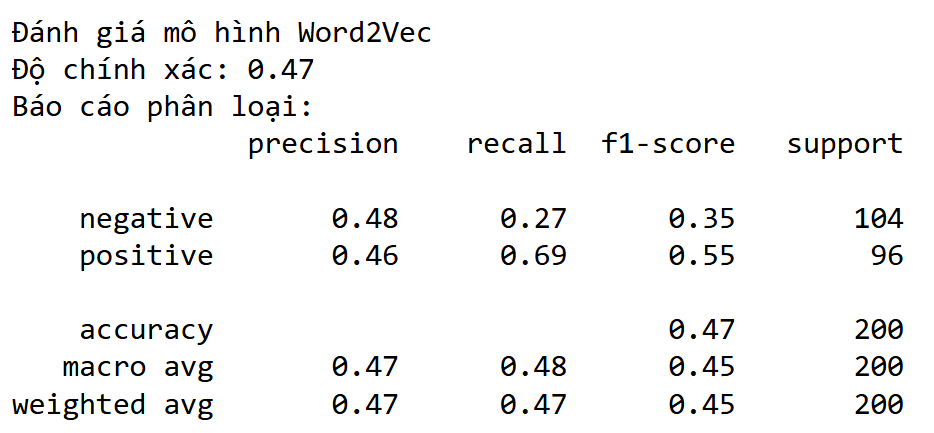


Hình 1.4.9 Chuẩn bị dữ liệu và mô hình phân loại SVM được chọn

Cuối cùng, đánh giá mô hình SVM đã được huấn luyện trên dữ liệu đã chuẩn bị. Độ chính xác của mô hình trên tập kiểm tra được tính toán bằng cách so sánh dự đoán của mô hình với nhãn thực tế. Báo cáo phân loại được in ra bao gồm các thông số như precision, recall và f1-score cho mỗi lớp (negative và positive).



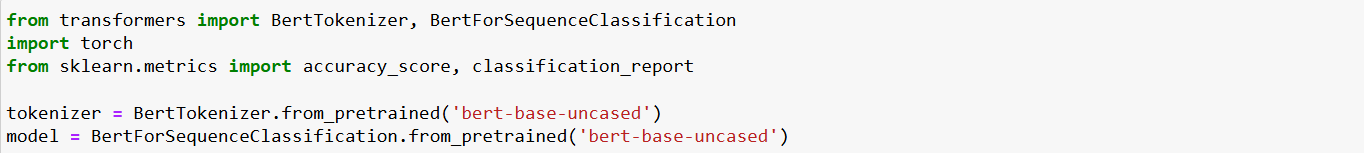
Hình 1.4.10 Đánh giá mô hình SVM



Hình 1.4.11 Kết quả đánh giá mô hình SVM

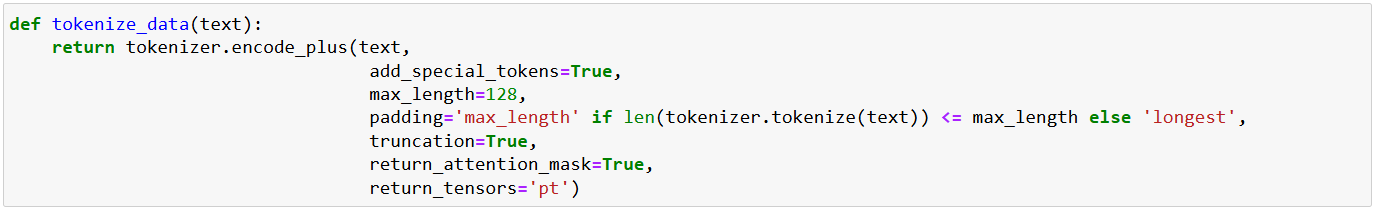
1.4.3.2 Mô hình 2: Sử dụng BPE

BERT được sử dụng trực tiếp để thực hiện các phần quan trọng của quy trình huấn luyện và đánh giá mô hình. Mã hóa từ vựng (bao gồm BPE) được sử dụng thông qua việc sử dụng BertTokenizer từ thư viện transformers.



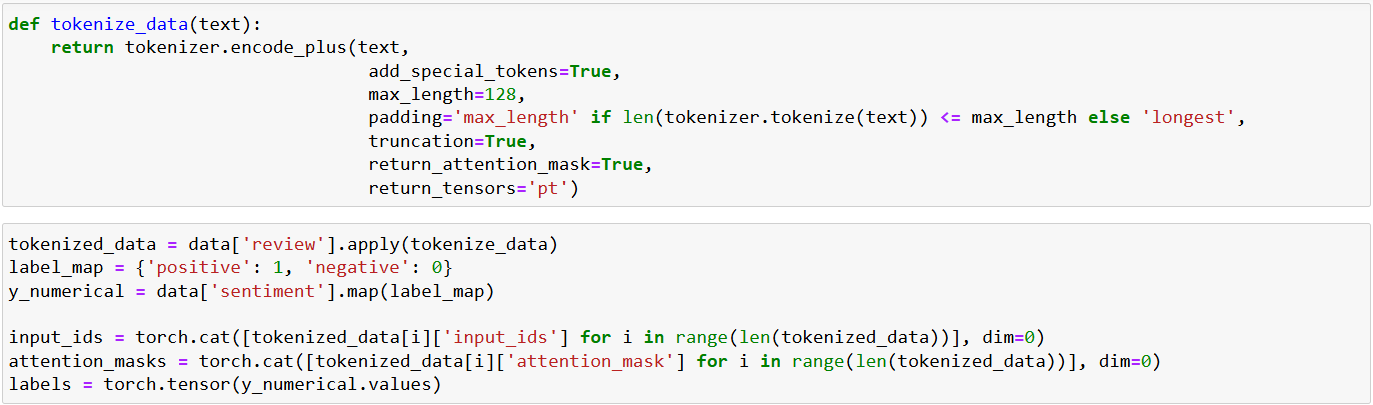
Hình 1.4.12 Tokenized dữ liệu cho mô hình BERT

Đầu tiên, chúng ta cần tokenize dữ liệu đầu vào bằng cách xây dựng hàm tokenize\_data sử dụng tokenizer.encode\_plus để mã hóa các câu thành các token bằng cách sử dụng BPE. Cụ thể, nó sử dụng tokenizer để mã hóa văn bản, thêm các token đặc biệt (CLS và SEP), giới hạn độ dài tối đa là 128, và trả về tensors.



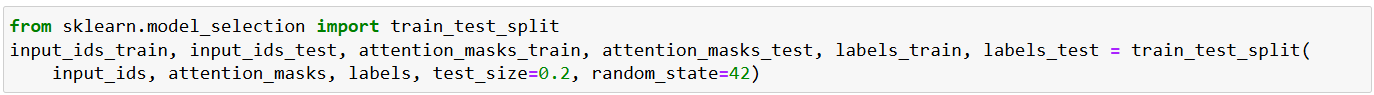
Hình 1.4.13 xây dựng hàm tokenize\_data

Dữ liệu văn bản (‘review’) được mã hóa bằng cách sử dụng hàm tokenize\_data và ánh xạ nhãn từ văn bản sang số (positive/negative ⭢ 1/0). Kết quả cuối cùng là các tensors input\_ids, attention\_masks, và labels, sẵn sàng cho việc huấn luyện mô hình.



Hình 1.4.14 Mã hóa và ánh xạ nhãn từ văn bản sang số

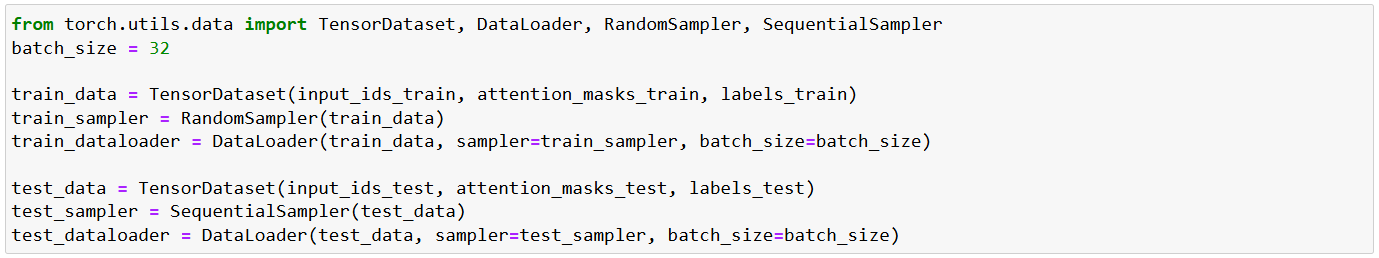
Sử dụng train\_test\_split từ thư viện sklearn để chia dữ liệu thành tập huấn luyện và tập kiểm tra. Tập kiểm tra chiếm 20% tổng số dữ liệu và sử dụng random\_state = 42 để đảm bảo tính nhất quán trong việc xáo trộn dữ liệu.



Hình 1.4.15 Chuẩn bị dữ liệu cho mô hình BERT

Sử dụng TensorDataset và DataLoader từ thư viện torch.utils.data để tạo DataLoader cho cả tập huấn luyện và tập kiểm tra giúp quản lý dữ liệu và tiến trình huấn luyện một cách hiệu quả.

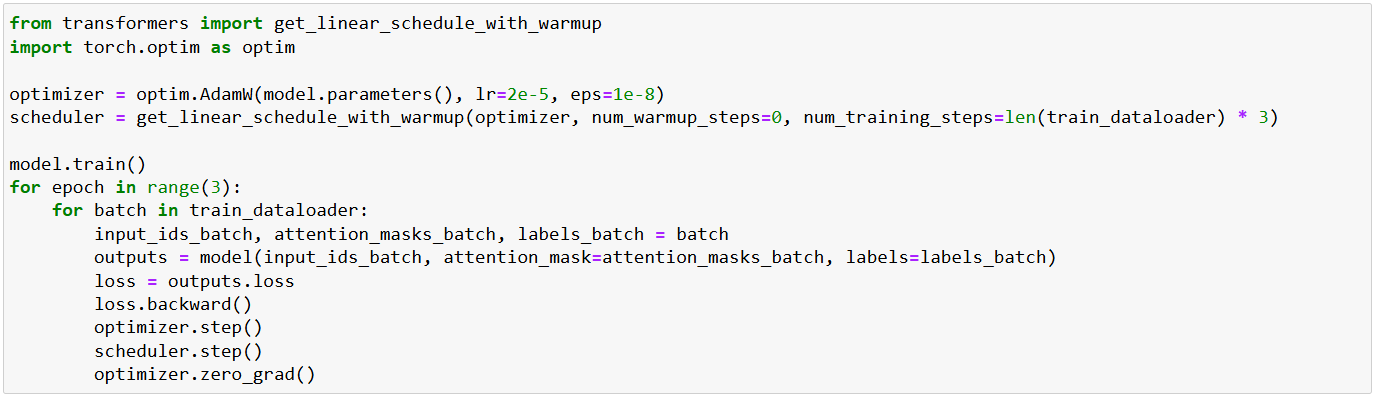
Sử dụng RandomSampler cho tập huấn luyện để mẫu được lấy ngẫu nhiên, trong khi SequentialSampler cho tập kiểm tra để các mẫu được lấy theo thứ tự.



Hình 1.4.16 T ạo DataLoader cho mô hình BERT

Sau khi chuẩn bị dữ liệu và tạo DataLoader, quá trình huấn luyện mô hình được thực hiện trong một số lượng epoch đã được chỉ định. Mỗi epoch, các batch dữ liệu được lấy từ train\_dataloader và sử dụng để tính toán loss và cập nhật trọng số của mô hình thông qua lan truyền ngược.

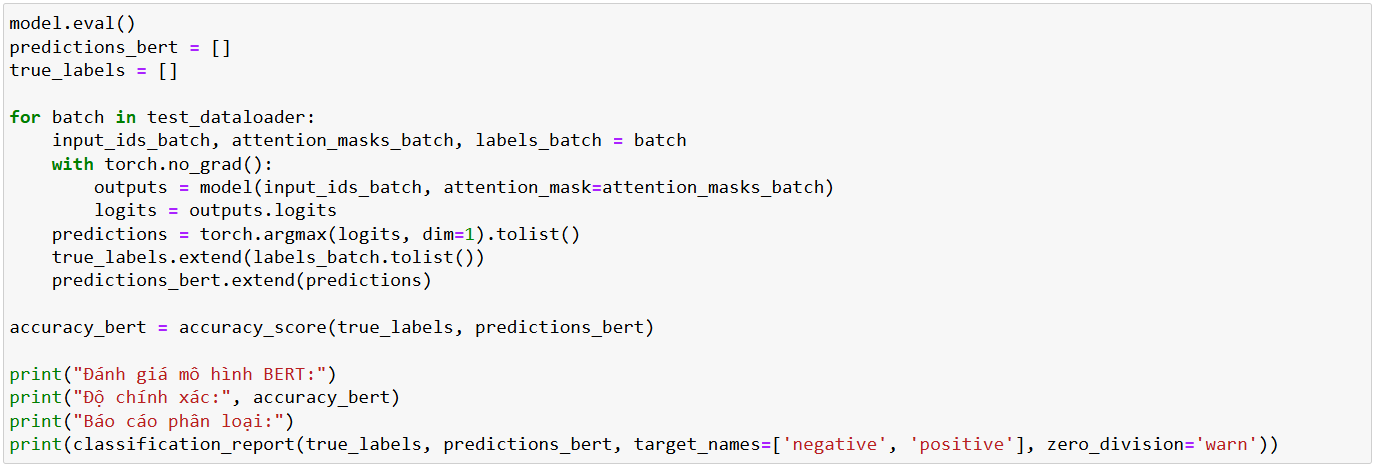
Để tối ưu hóa mô hình, chúng ta sử dụng thuật toán tối ưu hóa AdamW và áp dụng lên lịch lượng giảm tuyến tính với tăng nhiệt (linear schedule with warmup) để điều chỉnh tốc độ học theo thời gian.



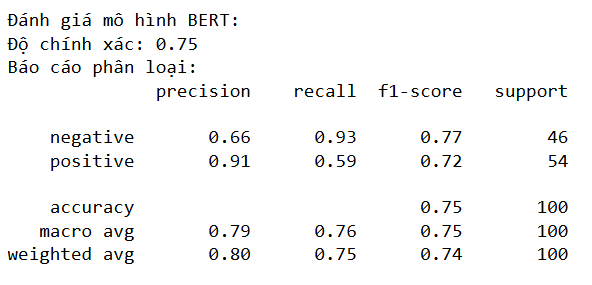
Hình 1.4.17 Huấn luyện mô hình BERT

Cuối cùng, đánh giá mô hình BERT đã được huấn luyện trên dữ liệu đã chuẩn bị. Các dự đoán được thực hiện cho từng batch trong test\_dataloader. logits được tính bằng cách truyền dữ liệu qua mô hình trong trạng thái đánh giá (eval) và không cập nhật các gradient (torch.no\_grad()).

Dự đoán cuối cùng được tính bằng cách lấy chỉ số có xác suất cao nhất. Độ chính xác của mô hình trên tập kiểm tra được tính toán bằng cách so sánh dự đoán của mô hình với nhãn thực tế. Báo cáo phân loại được in ra bao gồm các thông số như precision, recall và f1-score cho mỗi lớp (negative và positive).



Hình 1.4.18 Đánh giá mô hình BERT



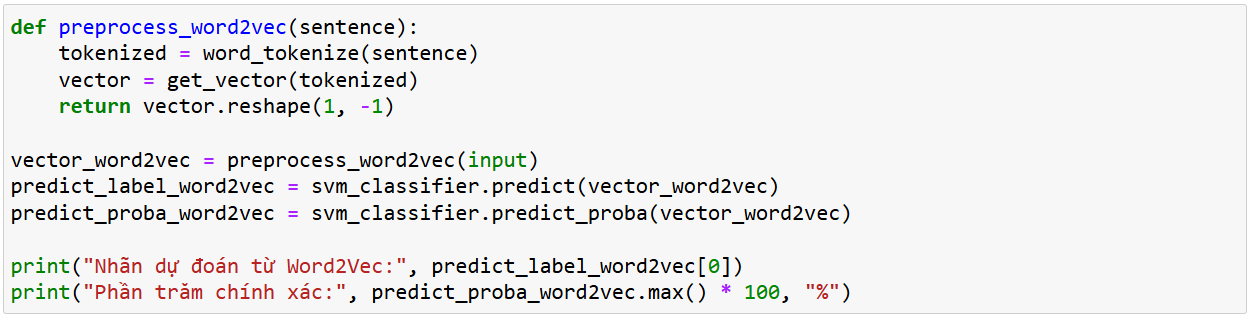
Hình 1.4.19 Kết quả đánh giá mô hình BERT

***1.4.4 Phân loại nhãn cho một review mới***

Phần này nhằm đánh giá hiệu suất của hai mô hình phân loại: Word2Vec và BERT trong việc phân loại đánh giá phim trên tập dữ liệu IMDB.

Cụ thể, chúng ta sẽ đánh giá kết quả của mỗi mô hình khi áp dụng cho một câu đánh giá mới được nhập từ bàn phím. Ở bài báo cáo này, nội dung mới được thêm vào ở đây là câu: “A film very good”

Với mô hình Word2Vec, câu đánh giá được tiền xử lý bằng cách sử dụng hàm preprocess\_word2vec, bao gồm việc mã hóa câu thành một vector. Sau đó, vector được đưa vào mô hình máy học SVM (Support Vector Machine) để dự đoán nhãn. Nhãn dự đoán và phần trăm chính xác được in ra màn hình.

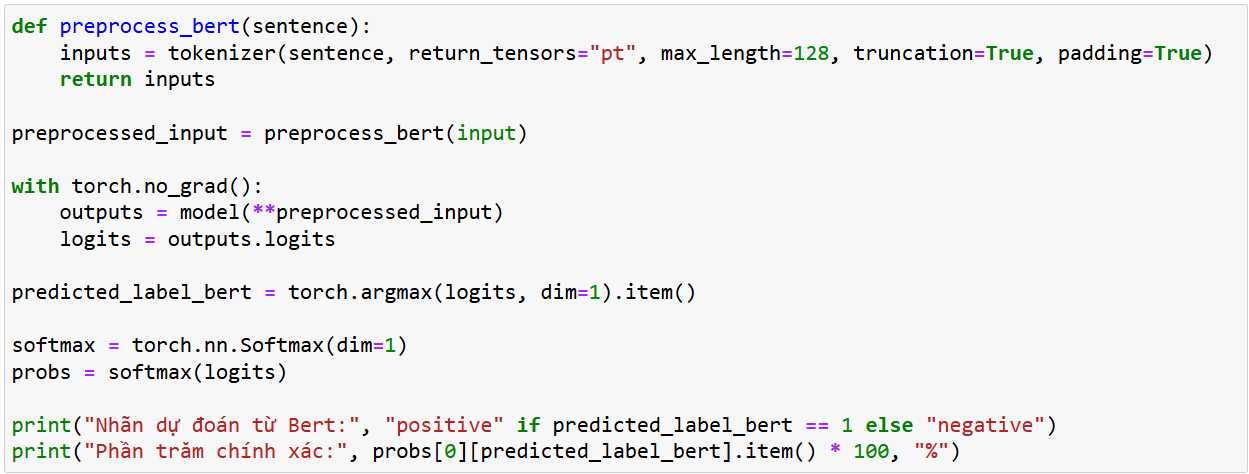


Hình 1.4.20 Xây dựng hàm preprocess\_word2vec



Hình 1.4.21 Kết quả in nhãn của mô hình Word2Vec

Với mô hình BERT, câu đánh giá cũng được tiền xử lý bằng cách sử dụng hàm preprocess\_bert, theo cách đưa vào BERT một cách phù hợp. Câu tiền xử lý sau đó được đưa vào mô hình BERT để dự đoán nhãn. Tính softmax được áp dụng để tính toán xác suất của mỗi nhãn. Nhãn dự đoán và phần trăm chính xác tương ứng được in ra màn hình.



Hình 1.4.22 Xây dựng hàm preprocess\_bert



Hình 1.4.23 Kết quả in nhãn của mô hình BERT

Dựa trên kết quả trên, mô hình BERT có hiệu suất tốt hơn trong việc phân loại đánh giá phim trên tập dữ liệu IMDB. Mặc dù cả hai mô hình đều dự đoán nhãn là Negative cho câu “A film very good”. Tuy nhiên, mô hình BERT có phần trăm chính xác cao hơn (68.03%) so với mô hình Word2Vec (51.35%).

Điều này cho thấy sự ưu việt của mô hình BERT trong việc hiểu biểu cảm và ngữ cảnh của câu, làm cho nó trở thành lựa chọn hiệu quả hơn trong các tác vụ phân loại đánh giá phim.

**CHƯƠNG 2 – BÀI TOÁN PHÁT HIỆN VÀ SỬA CHỮA LỖI CHÍNH TẢ TIẾNG VIỆT**

**2.1 Sơ lược về Seq2seq**

2.1.1 Định nghĩa

Mô hình seq2seq (Sequence to Sequence), hay còn gọi là mô hình mã hóa-giải mã, là một kiến trúc mạng nơ-ron được sử dụng để xử lý các chuỗi đầu vào và tạo ra chuỗi đầu ra tương ứng. Nó được ứng dụng rộng rãi trong các bài toán xử lý ngôn ngữ tự nhiên (NLP) như dịch máy, tóm tắt văn bản, trả lời câu hỏi tự nhiên, v.v.

Đặc điểm chính:

* Kiến trúc mã hóa-giải mã: Gồm hai thành phần chính: bộ mã hóa và bộ giải mã.
* Bộ mã hóa: Chuyển đổi chuỗi đầu vào thành một biểu diễn vectơ, thể hiện ngữ nghĩa của chuỗi.
* Bộ giải mã: Sử dụng vectơ ngữ nghĩa từ bộ mã hóa để tạo ra chuỗi đầu ra.
* Học tập dựa trên dữ liệu: Seq2seq được học bằng cách tối ưu hóa hàm mất mát trên tập dữ liệu huấn luyện.
* Khả năng xử lý chuỗi bất kỳ độ dài: Seq2seq có thể xử lý chuỗi đầu vào và đầu ra với độ dài khác nhau.

**2.2 Các hoạt động trong mã hóa và giải mã**

2.1.1 Mã hóa

Bộ mã hóa nhận chuỗi đầu vào và xử lý từng từ một.

Mỗi từ được biểu diễn bằng một vectơ nhúng (embedding vector).

Các vectơ nhúng được đưa qua mạng nơ-ron tuần hoàn (RNN) như LSTM hoặc GRU để thu được vectơ ngữ nghĩa.

Vectơ ngữ nghĩa cuối cùng được lưu trữ làm trạng thái của bộ mã hóa.

2.1.2 Giải mã

Bộ giải mã bắt đầu với một ký tự khởi đầu (thường là "<SOS>").

Trạng thái của bộ mã hóa được truyền vào bộ giải mã.

Bộ giải mã sử dụng trạng thái và đầu ra trước đó để dự đoán từ tiếp theo.

Từ dự đoán được thêm vào chuỗi đầu ra và trở thành đầu vào cho bước tiếp theo.

Quá trình lặp lại cho đến khi tạo ra ký tự kết thúc (thường là "</EOS>").

**2.3 Mô hình ứng dụng**

Dịch máy: Seq2seq là mô hình phổ biến nhất cho dịch máy tự động. Nó có thể học các mối quan hệ phức tạp giữa các ngôn ngữ và tạo ra bản dịch chất lượng cao.

Tóm tắt văn bản: Seq2seq có thể được sử dụng để tóm tắt văn bản dài thành một đoạn văn ngắn hơn, giữ nguyên ý nghĩa chính của văn bản gốc.

Trả lời câu hỏi tự nhiên: Seq2seq có thể được sử dụng để trả lời các câu hỏi tự nhiên bằng cách tạo ra câu trả lời phù hợp từ một đoạn văn bản hoặc cơ sở dữ liệu kiến thức.

Chát bot: Seq2seq có thể được sử dụng để xây dựng chatbot có thể giao tiếp với con người một cách tự nhiên và thông minh.

Tạo tiêu đề: Seq2seq có thể được sử dụng để tạo tiêu đề tự động cho các bài báo, video, v.v.

Phân tích tình cảm: Seq2seq có thể được sử dụng để phân tích cảm xúc của văn bản, xác định xem văn bản có tích cực, tiêu cực hay trung lập hay không.

Ngoài những ứng dụng trên, seq2seq còn có thể được sử dụng cho nhiều nhiệm vụ NLP khác như tạo văn bản sáng tạo, nhận diện thực thể tên, v.v.

**2.4 Ứng dụng vào xây dựng bài toán**

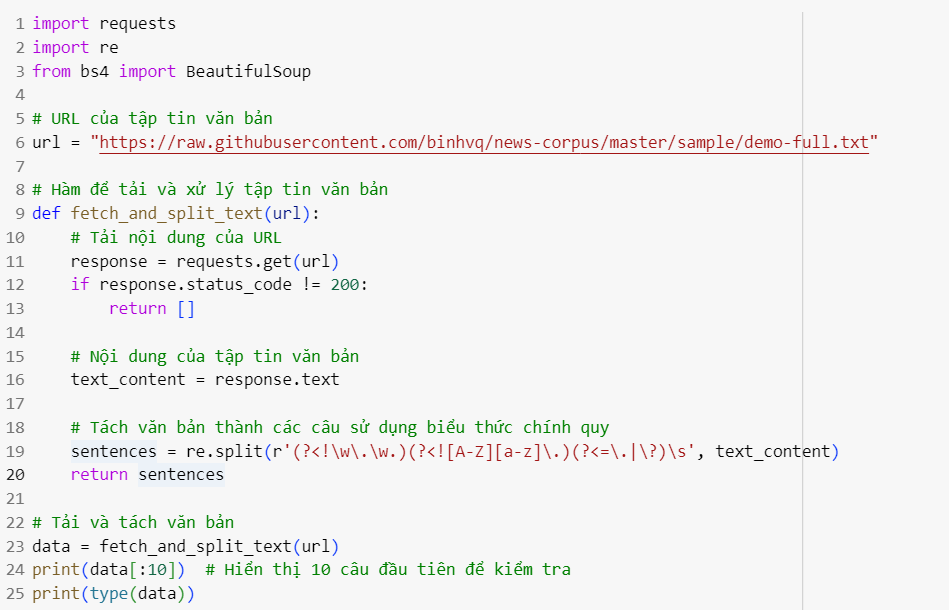
***2.4.1 Mô tả bài toán***

Mục tiêu của bài toán là xây dựng chương trình phát hiện và sửa chữa lỗi chính tả tiếng Việt theo tiếp cận học sâu.

***2.4.2. Xây dựng bài toán***

***2.4.2.1 Chuẩn bị data***

Để xây dựng được bài toán phát hiện và sửa lỗi chính tả, việc đầu tiên cần làm là chuẩn bị dữ liệu cho bài toán, cụ thể việc này gồm link url dẫn đến dữ liệu và import thư viện cần thiết để lấy dữ liệu từ link url đó.



Hình ảnh 2.4.1 Chuẩn bị data

Trong đoạn code trên, dữ liệu được lấy từ link url và tải xuống thông qua hàm “fetch\_and\_split\_text”.

Hàm “fetch\_and\_split\_text”:

* Hàm này nhận một URL làm đối số.
* Nó sử dụng thư viện requests để tải nội dung của URL.
* Nếu mã trạng thái của phản hồi không phải là 200 (thành công), nó sẽ trả về một danh sách rỗng.
* Nếu tải xuống thành công, nó sẽ trích xuất nội dung văn bản từ phản hồi.
* Nó sử dụng biểu thức chính quy để chia tách văn bản thành các câu.
* Biểu thức chính quy r'(?<!\w\.\w.)(?<! [A-Z][a-z]\.)(?<=\.|\?)\s' được sử dụng để tìm các dấu chấm và dấu hỏi theo sau là dấu cách, nhưng không phải là dấu chấm sau một chữ cái và dấu chấm hoặc dấu chấm sau một chữ cái viết hoa và chữ cái viết thường.
* Hàm này trả về một danh sách các câu.

Sau khi thực hiện hàm trên, dữ liệu trả về sẽ lưu trong “data”, 2 dòng print ở cuối dùng để kiểm tra xem liệu dữ liệu đã được tải thành công chưa và để xem dữ liệu mẫu.

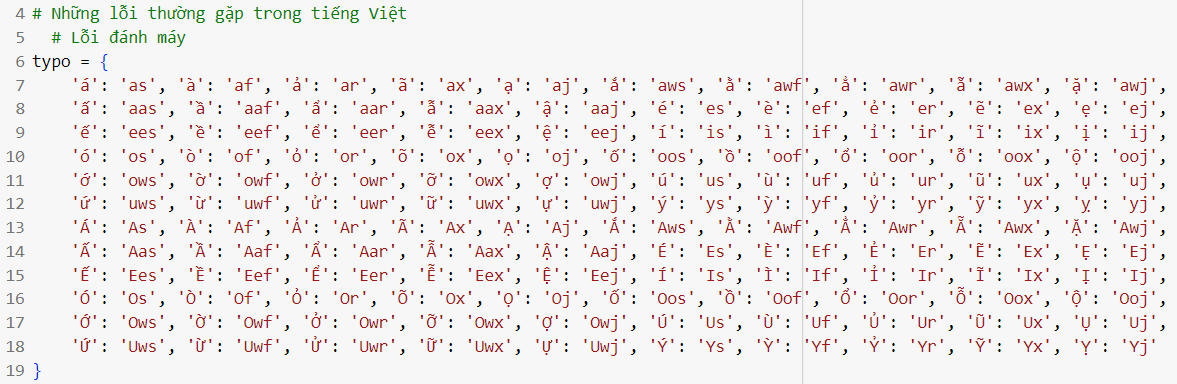
***2.4.2.2 Chuẩn bị danh sách các lỗi thường gặp***

Tiếp theo là một bước quan trọng cho bài toán, để có thể xây dựng bài toán phát hiện và sửa chữa lỗi thì cần phải chuẩn bị một danh sách chứa các lỗi thường gặp trong tiếng Việt.

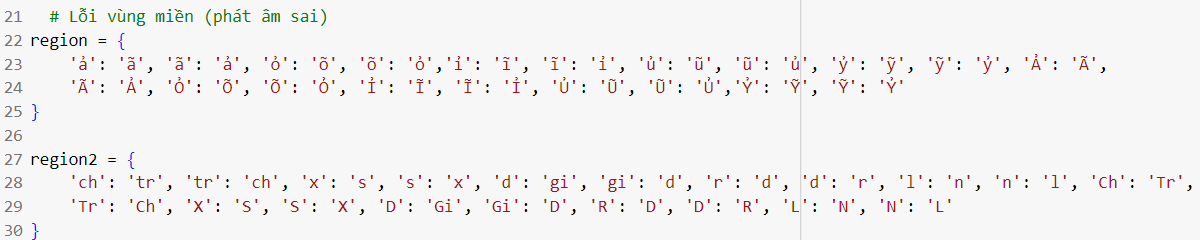
Đầu tiên là danh sách các nguyên âm và chữ cái có trong tiếng Việt.



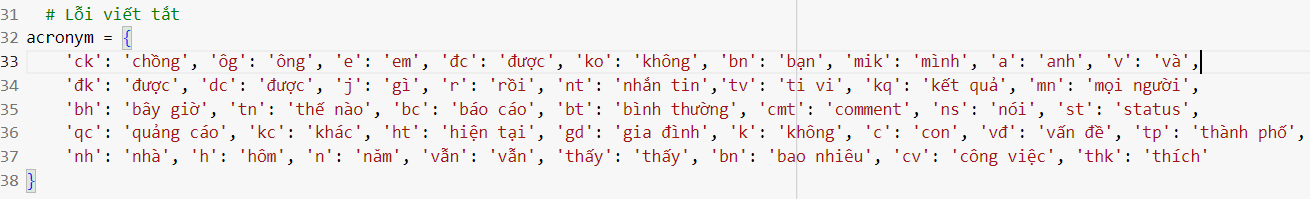
Tiếp theo là danh sách lỗi thường gặp trong tiếng Việt bao gồm lỗi đánh máy, lỗi vùng miền (phát âm sai) và lỗi viết tắt.



Hình ảnh 2.4.2 Danh sách lỗi đánh máy



Hình ảnh 2.4.3 Danh sách lỗi vùng miền

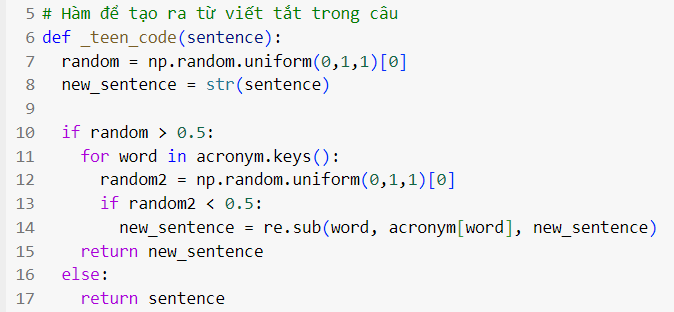


Hình ảnh 2.4.4 Danh sách lỗi viết tắt

Việc liệt kê các danh sách lỗi như thế này sẽ giúp việc huấn luyện mô hình dễ dàng hơn và chính xác hơn.

***2.4.2.3 Hàm phát sinh lỗi chính tả thường gặp***

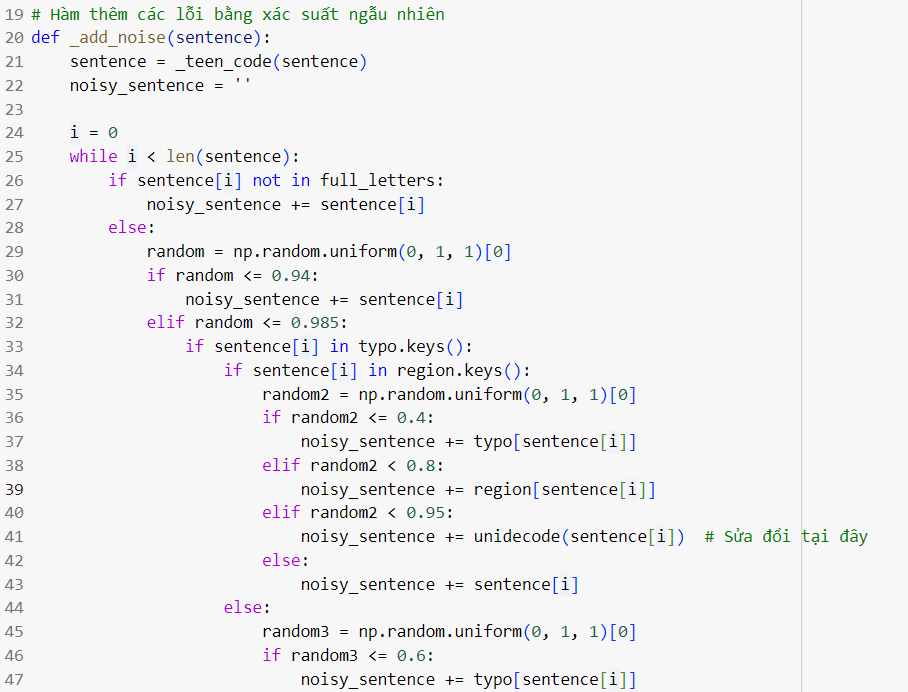
Nhiệm vụ chính của phần này là tạo ra lỗi chính tả thường gặp ví dụ như teencode và một số lỗi khác.



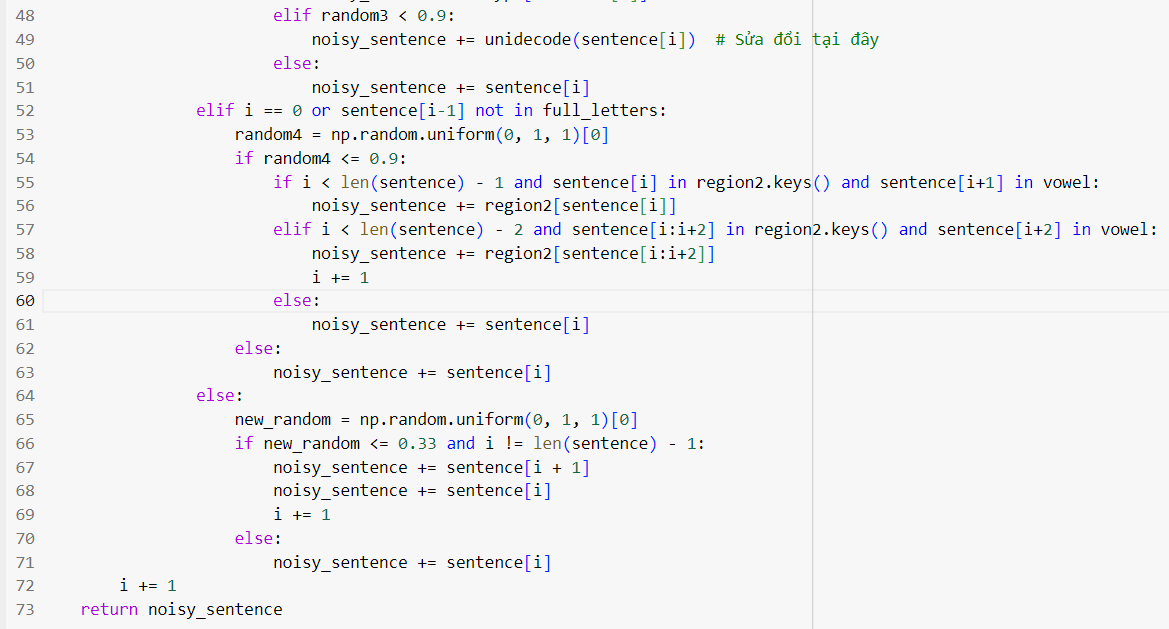
Hình ảnh 2.4.5 Hàm \_teen\_code

Hàm “\_teen\_code”:

* Hàm này nhận một câu văn bản làm đối số.
* Sử dụng thư viện numpy để tạo một số ngẫu nhiên giữa 0 và 1.
* Nếu số ngẫu nhiên lớn hơn 0.5, hàm sẽ duyệt qua một từ điển acronym (chưa được định nghĩa trong đoạn code này).
* Trong quá trình lặp, nó lại tạo ra một số ngẫu nhiên khác.
* Nếu số ngẫu nhiên thứ hai nhỏ hơn 0.5, hàm sẽ thay thế các từ trong câu với các từ viết tắt tương ứng từ từ điển acronym.
* Cuối cùng, hàm trả về câu văn bản đã được thay thế từ viết tắt (nếu có).



Hình ảnh 2.4.6 Hàm \_add\_noise (p1)

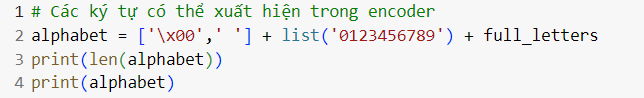


Hình ảnh 2.4.7 Hàm \_add\_noise (p2)

Hàm “\_add\_noise”:

* Hàm này nhận một câu văn bản làm đối số.
* Đầu tiên, nó gọi hàm \_teen\_code để có khả năng thay đổi câu bằng từ viết tắt.
* Sau đó, nó khởi tạo một chuỗi ký tự rỗng noisy\_sentence để lưu trữ câu văn bản bị lỗi.
* Nó lặp qua từng ký tự trong câu văn bản ban đầu.
* Nếu ký tự không phải là chữ cái, nó sẽ được thêm trực tiếp vào chuỗi noisy\_sentence
* Nếu ký tự là chữ cái, hàm lại tạo ra một số ngẫu nhiên khác.
* Dựa trên giá trị của số ngẫu nhiên, hàm có thể thực hiện các thao tác sau:
* Thêm chính xác ký tự đó vào chuỗi noisy\_sentence (xác suất 94%).
* Thay thế bằng ký tự gõ nhầm từ một từ điển typo (nếu có, xác suất 4.5%).
* Thay thế bằng một ký tự thay thế theo vùng miền từ một từ điển region (nếu có và tùy theo xác suất).
  + Tại đây có một sửa đổi: Nếu không thay bằng ký tự vùng miền, hàm có thể thay thế bằng phiên bản unidecode của ký tự đó (xác suất 5%).
* Thay thế bằng phiên bản unidecode của ký tự đó (nếu không thuộc các trường hợp trên, xác suất 1.5%).
* Ngoài ra, hàm còn xử lý các trường hợp đặc biệt khác như:
* Kiểm tra vị trí ký tự (đầu câu hoặc sau ký tự không phải chữ cái) để quyết định thay thế sang ký tự vùng miền kép (từ một từ điển region2, nếu có).
* Sao chép nhầm ký tự tiếp theo (xác suất 33%).
* Cuối cùng, hàm trả về chuỗi ký tự noisy\_sentence chứa câu văn bản đã được thêm lỗi.

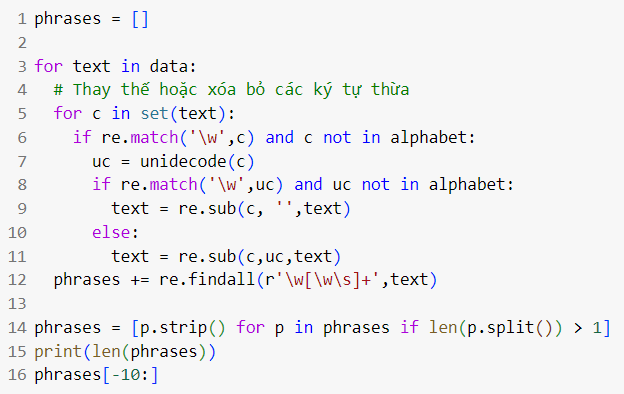
Đoạn code sau để xác định ký tự có thể xuất hiện trong encoder



Hình ảnh 2.4.8 Ký tự trong encoder

***2.4.2.4 Tách câu thành các cụm từ***

Nhiệm vụ chính của phần này là tách các câu trong data thành các cụm từ.



Hình ảnh 2.4.9 Tách câu thành cụm từ

Trong đoạn code trên, phrases được tạo ra để lưu trữ các cụm từ sau khi xử lý dữ liệu.

Vòng lặp này lặp qua từng chuỗi văn bản trong tập dữ liệu data. Đối với mỗi chuỗi văn bản, mã thực hiện các bước sau:

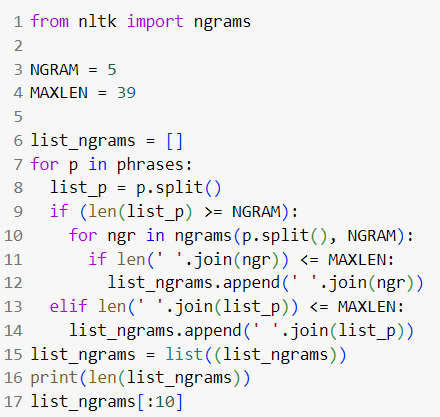
* Thay thế hoặc xóa bỏ các ký tự thừa:

Mã sử dụng biểu thức chính quy re.match('\w',c) để kiểm tra xem ký tự c có phải là chữ cái hay không.

* Nếu ký tự c là chữ cái và không thuộc bảng chữ cái alphabet, mã sử dụng hàm unidecode để chuyển đổi ký tự c sang ký tự Unicode tương ứng.
* Mã sử dụng hàm re.sub để thay thế tất cả các lần xuất hiện của ký tự c trong chuỗi văn bản bằng ký tự Unicode tương ứng.
* Nếu ký tự c không phải là chữ cái hoặc thuộc bảng chữ cái alphabet, mã sử dụng hàm re.sub để xóa bỏ tất cả các lần xuất hiện của ký tự c trong chuỗi văn bản.
* Tách văn bản thành các cụm từ:
* Mã sử dụng biểu thức chính quy r'\w\w\s]+' để tìm kiếm tất cả các cụm từ gồm hai hoặc nhiều từ trong chuỗi văn bản.
* Mã sử dụng hàm re.findall để tìm tất cả các cụm từ khớp với biểu thức chính quy và lưu trữ chúng vào danh sách phrases.
* Loại bỏ các cụm từ chỉ chứa một từ:
* Mã sử dụng danh sách comprehension để lọc danh sách phrases chỉ giữ lại các cụm từ có độ dài lớn hơn 1.
* Hàm strip() được sử dụng để loại bỏ các khoảng trắng thừa ở đầu và cuối mỗi cụm từ.

***2.4.2.5 Tách các cụm từ thành danh sách 5-grams với MAXLEN =39***

Nhiệm vụ chính của phần này tách các cụm từ thành danh sách các cụm 5-grams với maxlen là 39.



Hình ảnh 2.4.10 Tách các cụm từ thành danh sách 5-grams với maxlen là 39.

Import thư viện ngrams từ thư viện nltk. Thư viện này cung cấp các hàm để tạo ra các n-gram từ văn bản.

Khai báo biến NGRAM để lưu trữ giá trị của n, tức là độ dài của n-gram. Trong trường hợp này, NGRAM được đặt thành 5, nghĩa là n-gram sẽ được tạo ra với độ dài 5 từ.

Khai báo biến MAXLEN để lưu trữ giá trị độ dài tối đa của n-gram. Trong trường hợp này, MAXLEN được đặt thành 39, nghĩa là chỉ những n-gram có độ dài tối đa 39 từ mới được tạo ra.

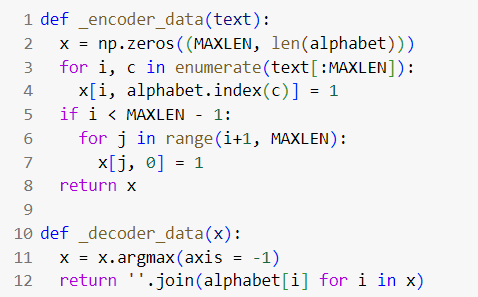
Vòng lặp này lặp qua từng cụm từ trong danh sách phrases. Đối với mỗi cụm từ, mã thực hiện các bước sau:

* Kiểm tra độ dài của cụm từ:
  + Nếu độ dài của cụm từ lớn hơn hoặc bằng NGRAM, mã chia cụm từ thành danh sách các từ và lặp qua từng n-gram có thể được tạo ra từ danh sách các từ đó.
  + Nếu n-gram có độ dài tối đa nhỏ hơn hoặc bằng MAXLEN, mã thêm n-gram vào danh sách list\_ngrams.
* Nếu cụm từ có độ dài nhỏ hơn MAXLEN thì mã thêm vào cùm từ danh sách list\_ngrams

Dòng 15 dùng để chuyển đổi danh sách list\_ngrams thành tuple.

***2.4.2.6 Hàm encoder và decoder***

Nhiệm vụ chính của phần này là tạo dữ liệu cho encoder và decoder.



Hình ảnh 2.4.11 Hàm \_encoder\_data và \_decoder\_data

Hàm “\_encoder\_data”:

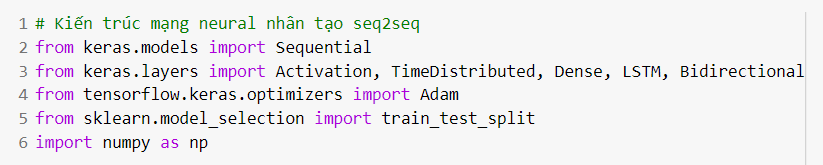
* Hàm này mã hóa một chuỗi ký tự thành một ma trận nhị phân.
* Ma trận nhị phân có kích thước (MAXLEN, len(alphabet)).
* Hàng thứ i của ma trận nhị phân tương ứng với ký tự thứ i của chuỗi đầu vào.
* Cột thứ j của ma trận nhị phân tương ứng với ký tự thứ j trong bảng chữ cái.
* Giá trị của phần tử (i, j) trong ma trận nhị phân là 1 nếu ký tự thứ i của chuỗi đầu vào bằng ký tự thứ j trong bảng chữ cái, và 0 nếu không.

Hàm “\_encoder\_data”:

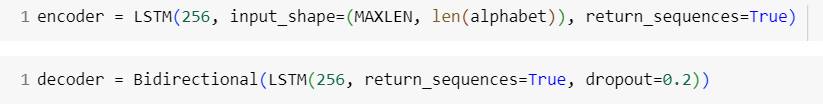
* Hàm này giải mã một ma trận nhị phân thành một chuỗi ký tự.
* Hàm này sử dụng hàm argmax để tìm vị trí của giá trị lớn nhất trong mỗi cột của ma trận nhị phân.
* Vị trí này tương ứng với ký tự trong bảng chữ cái được sử dụng để mã hóa ký tự tương ứng trong chuỗi đầu vào.
* Hàm này sử dụng vòng lặp để kết nối các ký tự được giải mã thành một chuỗi.

***2.4.2.7 Xây dựng mô hình***

Nhiệm vụ chính của phần này là xây dựng mô hình encoder và decoder dựa trên kiến trúc mạng neural nhân tạo seq2seq.



Hình ảnh 2.4.11 Import thư viện cần thiết cho xây dựng model



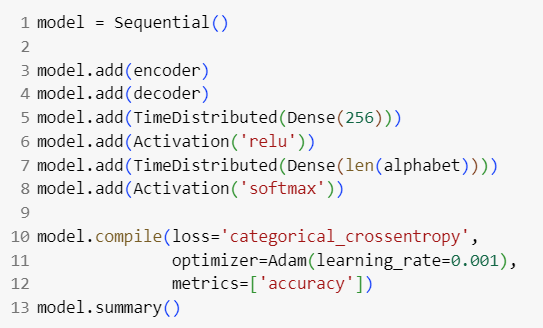
Hình ảnh 2.4.12 Xây dựng encoder và decoder

Mô hình encoder:

* 256: Kích thước của vector ẩn cho mỗi đơn vị LSTM.
* (MAXLEN, len(alphabet)): Hình dạng của đầu vào.
* MAXLEN: Chiều dài tối đa của chuỗi đầu vào.
* len(alphabet)\*\*: Số lượng ký tự trong bảng chữ cái.
* return\_sequences=True: Cho biết mô hình nên trả về chuỗi đầu ra hay không. Trong trường hợp này, mô hình sẽ trả về chuỗi đầu ra có cùng chiều dài với chuỗi đầu vào.

Mô hình decoder:

* Bidirectional: Biến đổi mô hình LSTM thành mô hình LSTM hai chiều.
* LSTM(256, return\_sequences=True, dropout=0.2): Mô hình LSTM hai chiều với 256 đơn vị ẩn.
* return\_sequences=True: Cho biết mô hình nên trả về chuỗi đầu ra hay không.
* dropout=0.2: Tỷ lệ dropout được áp dụng cho đầu ra của mỗi đơn vị LSTM.



Hình ảnh 2.4.13 Thêm các layer vào model

Sau khi tạo một model sử dụng mô hình tuần tự (Sequential) thì lần lượt add các layers là encoder và decoder vào model. Sau đó thêm một lớp TimeDistributed để phân phối các lớp Dense trên mỗi lần lặp thời gian.Tiếp tục thêm một lớp Dense để tạo ra số lượng ký tự trong bảng chữ cái. Sau cùng biên dịch model bằng hàm compile() và in tóm tắt mô hình bằng hàm summary()

***2.4.2.8 Tách dữ liệu thành tập training và validation***

Nhiệm vụ chính của phần này là chia dữ liệu thành tập training và tập validation.



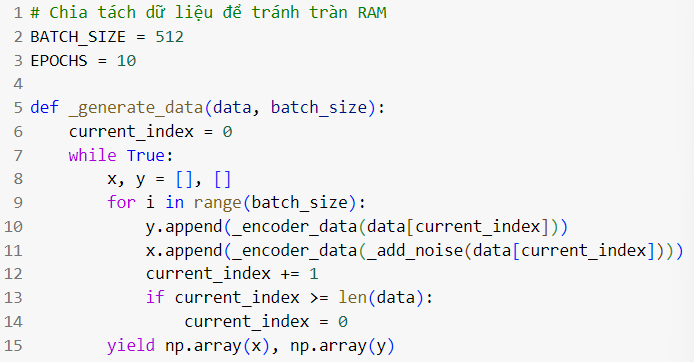
Hình ảnh 2.4.14 Chia dữ liệu

Câu lệnh này chia tách tập dữ liệu list\_ngrams thành hai tập con: train\_data và valid\_data.

* list\_ngrams là một danh sách các n-gram. N-gram là một chuỗi liên tiếp gồm n từ trong một văn bản.
* test\_size=0.2 cho biết tỷ lệ dữ liệu được sử dụng để tạo ra tập dữ liệu xác nhận. Trong trường hợp này, 20% dữ liệu được sử dụng để tạo ra tập dữ liệu xác nhận.
* random\_state=42 là một số nguyên được sử dụng để đảm bảo rằng việc chia tách dữ liệu luôn giống nhau mỗi khi chương trình được chạy.

Biến train\_data lưu trữ tập dữ liệu huấn luyện. Tập dữ liệu huấn luyện được sử dụng để huấn luyện mô hình học máy.

Biến valid\_data lưu trữ tập dữ liệu xác nhận. Tập dữ liệu xác nhận được sử dụng để đánh giá hiệu suất của mô hình học máy.



Hình ảnh 2.4.15 Hàm \_generate\_data

Khai báo hằng số BATCH\_SIZE với giá trị là 512. Đây là kích thước của mỗi batch dữ liệu.

Khai báo hằng số EPOCHS với giá trị là 10. Đây là số lần lặp lại toàn bộ dữ liệu trong quá trình huấn luyện mô hình.

Hàm \_generate\_data nhận hai tham số: data là tập dữ liệu và batch\_size là kích thước batch.

Hàm này sẽ lặp lại vô thời hạn cho đến khi có yêu cầu dừng.

Trong mỗi vòng lặp, hàm này sẽ tạo một batch dữ liệu mới có kích thước batch\_size.

Mỗi batch dữ liệu bao gồm hai phần:

* Dữ liệu đầu vào: Dữ liệu đầu vào được mã hóa bằng hàm \_encoder\_data.
* Dữ liệu đầu ra: Dữ liệu đầu ra là dữ liệu gốc mà không cần mã hóa.

Hàm này sẽ sử dụng câu lệnh yield để trả về một batch dữ liệu dưới dạng mảng NumPy.

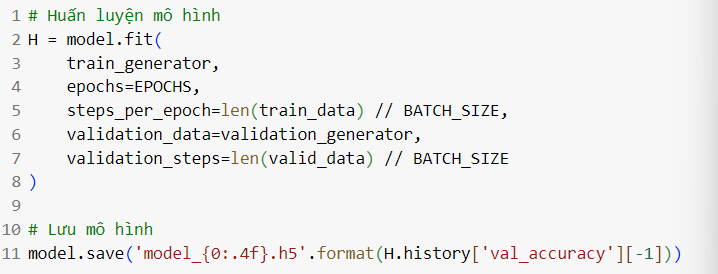


Hình ảnh 2.4.16 Tạo train\_generator và validation\_generator

Đoạn code trên dùng để tạo data cho train và validation bằng cách sử dụng hàm \_generate\_data() ở trên và tạo tập data có batch\_size là 512.

***2.4.2.9 Huấn luyện mô hình***

Nhiệm vụ chính của phần này là huấn luyện mô hình để nó thực hiện nhiệm vụ phát hiện và sửa lỗi và lưu nó thành model để sử dụng sau này.



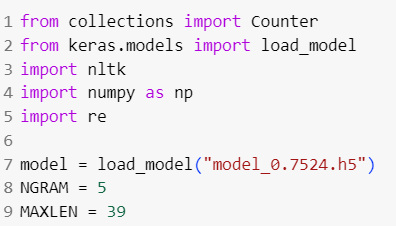
Hình ảnh 2.4.17 Huấn luyện model

Trong đoạn code trên:

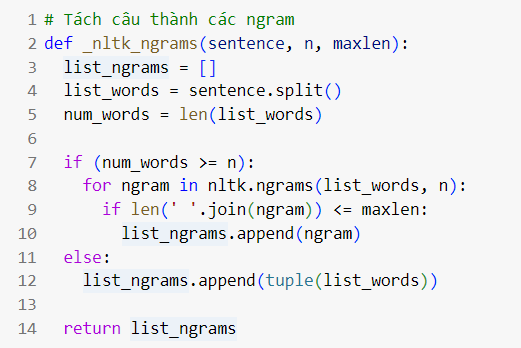
* Biến EPOCHS xác định số lần lặp lại quá trình huấn luyện.
* Biến BATCH\_SIZE xác định số lượng mẫu được xử lý trong mỗi lần lặp lại.
* Biến train\_generator cung cấp dữ liệu huấn luyện cho mô hình.
* Biến validation\_generator cung cấp dữ liệu xác nhận cho mô hình.
* Biến validation\_steps xác định số lượng mẫu xác nhận được xử lý trong mỗi lần lặp lại.
* Mô hình được lưu vào tệp có tên là model\_{0:.4f}.h5
* Biến {0:.4f} định dạng giá trị độ chính xác xác nhận cuối cùng thành một chuỗi có độ chính xác bốn số thập phân.

***2.4.2.10 Ứng dụng mô hình***

Import các thư viện cần thiết và load model đã huấn luyện ở trên



Hình ảnh 2.4.18 Import thư viện cần thiết và load model



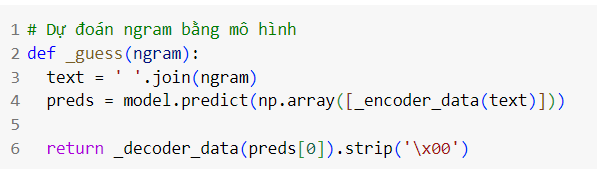
Hình ảnh 2.4.19 Hàm nltk\_ngrams

Hàm “\_nltk\_ngrams”:

* sentence: Câu cần tách thành các ngram
* n: Kích thước của ngram. Ví dụ: nếu n = 2, hàm sẽ tách câu thành các bigram.
* maxlen: Độ dài tối đa của ngram. Ví dụ: nếu maxlen = 10, hàm sẽ chỉ tách các ngram có độ dài tối đa là 10 ký tự.

Hàm \_nltk\_ngrams thực hiện các bước sau:

* + Tách câu thành các từ bằng hàm split().
  + Khởi tạo một danh sách rỗng để lưu trữ các ngram.
  + Tính số lượng từ trong câu.
  + Kiểm tra xem số lượng từ trong câu có lớn hơn hoặc bằng kích thước ngram hay không.
  + Nếu số lượng từ trong câu lớn hơn hoặc bằng kích thước ngram, sử dụng hàm nltk.ngrams để tạo các ngram có kích thước n từ câu.
  + Duyệt qua các ngram và kiểm tra xem độ dài của ngram có nhỏ hơn hoặc bằng độ dài tối đa hay không.
  + Nếu độ dài của ngram nhỏ hơn hoặc bằng độ dài tối đa, thêm ngram vào danh sách ngram.
  + Nếu độ dài của ngram lớn hơn độ dài tối đa, thêm toàn bộ câu vào danh sách ngram.
  + Trả về danh sách ngram.



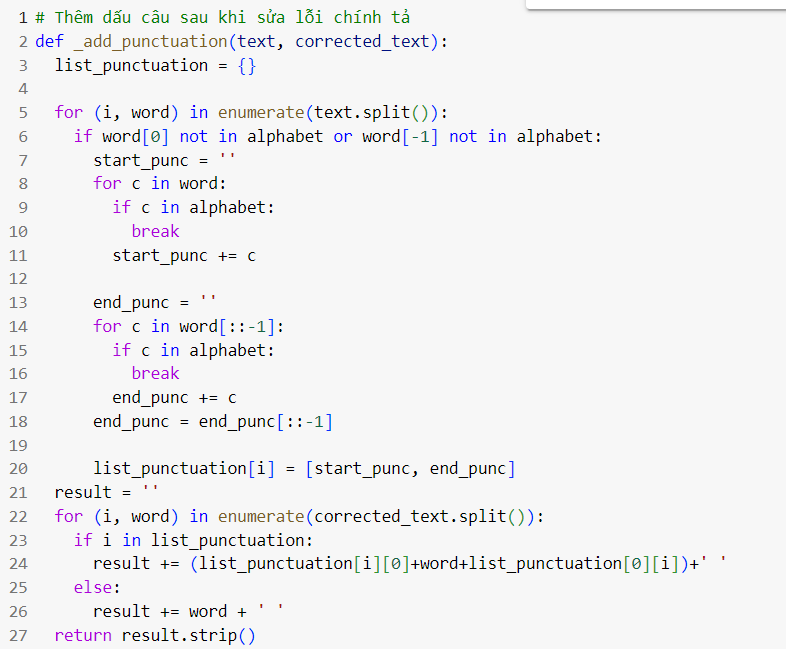
Hình ảnh 2.4.20 Hàm guess

Hàm “\_guess” nhận ba tham số đầu vào:

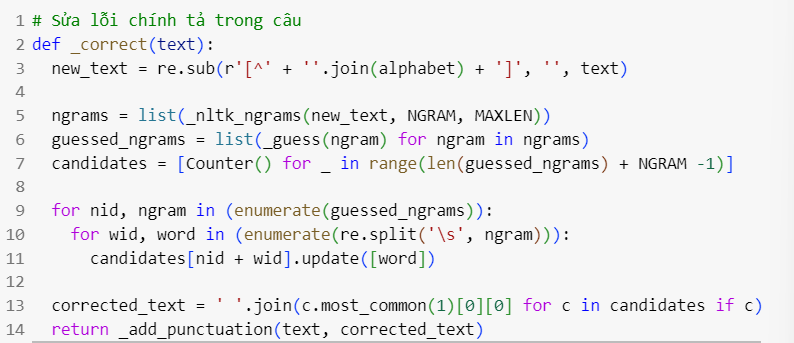
* ngram: Ngram hiện tại.
* model: Mô hình dự đoán ngram.
* encoder: Hàm mã hóa ngram thành vectơ đầu vào cho mô hình dự đoán.
* decoder: Hàm giải mã vectơ đầu ra của mô hình dự đoán thành ngram tiếp theo.

Hàm “\_guess” thực hiện các bước sau:

* Mã hóa ngram hiện tại thành vectơ đầu vào cho mô hình dự đoán.
* Dự đoán ngram tiếp theo bằng mô hình dự đoán.
* Giải mã vectơ đầu ra của mô hình dự đoán thành ngram tiếp theo.
* Trả về ngram tiếp theo.



Hình ảnh 2.4.21 Hàm \_add\_punctuation



Hình ảnh 2.4.22 Hàm \_correct

Hàm \_correct: Hàm này nhận một chuỗi văn bản làm đầu vào và trả về chuỗi văn bản được sửa lỗi chính tả.

Biến alphabet: Biến này chứa một chuỗi các ký tự đại diện cho bảng chữ cái tiếng Việt.

Biến NGRAM: Biến này xác định kích thước của ngram được sử dụng để sửa lỗi chính tả.

Biến MAXLEN: Biến này xác định độ dài tối đa của ngram.

Hàm \_nltk\_ngrams: Hàm này được sử dụng để tạo các ngram từ chuỗi văn bản.

Hàm \_guess: Hàm này được sử dụng để dự đoán từ chính xác cho một ngram.

Danh sách candidates: Danh sách này chứa các ứng cử viên cho từ chính xác cho mỗi ngram.

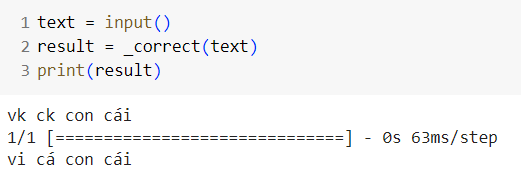
Vòng lặp for: Vòng lặp này duyệt qua các ngram và cập nhật danh sách candidates với các ứng cử viên từ chính xác.

Biến corrected\_text: Biến này chứa chuỗi văn bản được sửa lỗi chính tả.

Hàm \_add\_punctuation: Hàm này được sử dụng để thêm dấu chấm câu vào chuỗi văn bản.

Hàm “\_correct” thực hiện các bước sau:

* Chuỗi văn bản đầu vào được thay thế các ký tự không phải chữ cái bằng khoảng trắng.
* Các ngram được tạo từ chuỗi văn bản được sửa đổi.
* Mỗi ngram được dự đoán từ chính xác bằng cách sử dụng hàm \_guess.
* Các ứng cử viên từ chính xác cho mỗi ngram được thêm vào danh sách candidates.
* Chuỗi văn bản được sửa lỗi chính tả được tạo ra bằng cách kết hợp các từ chính xác nhất từ danh sách candidates.
* Dấu chấm câu được thêm vào chuỗi văn bản được sửa lỗi chính tả.



Hình ảnh 2.4.23 Test model

Cuối cùng test xem mô hình có hoạt động đúng mong đợi hay không.

**CHƯƠNG 3 – KẾT LUẬN**

**3.1 Kết quả đạt được**

Trong quá trình thực hiện, nhóm 10 đã đạt được các kết quả sau:

* Hiểu rõ khái niệm và đặc điểm của BPE, nắm được lợi ích của việc sử dụng BPE trong lĩnh vực xử lý ngôn ngữ tự nhiên.
* Xây dựng được các thuật toán sử dụng phương pháp BPE và không sử dụng phương pháp BPE.
* Hiểu biết thêm về encoder-decoder và ứng dụng nó vào việc xây dựng bài toán phát hiện và sửa chữa lỗi chính tả.

**3.2 Thuận lời và khó khăn**

***3.2.1 Thuận lợi***

* Có nguồn tài liệu phong phú, đa dạng.
* Thư viện hỗ trợ: Được sử dụng nhiều thư viện có sẵn để triển khai trên nhiều mô hình.
* Tích hợp với Python: một ngôn ngữ lập trình phổ biến và dễ học, giúp việc phát triển và triển khai dễ dàng hơn.

***3.2.2 Khó khăn***

* Phải học hỏi và hiểu biết về các khái niệm và cách thức hoạt động của các mô hình để sử dụng các mô hình hiệu quả,
* Triển khai nhiều mô hình cần phải sử dụng nhiều thư viện đòi hỏi kiến thức đủ tốt về Python.

**3.3 Đánh giá**

|  |  |  |
| --- | --- | --- |
| **Câu** | **Tự đánh giá** | **Độ hoàn thành** |
| 1 – Phương pháp BPE | Tốt | 100% |
| 2 – Xây dựng mô hình phát hiện và sửa lỗi chính tả | Tốt | 100% |

Bảng 4.3 Đánh giá độ hoàn thành đồ án

**TÀI LIỆU THAM KHẢO**

**Tiếng Việt**

1. *“BPE – Byte Pair Encoding – Vũ khí bí mật của NLP hiện đại”* Trí tuệ nhân tạo [Online].
2. Đ.V.D.Thanh, N.V.Quang, N.V.Cường, P.Đ.Khoa và L.K.H.Phúc (2020), *“*[*Xử lý Ngôn ngữ Tự nhiên: Tiền Huấn luyện*](https://d2l.aivivn.com/chapter_natural-language-processing-pretraining/word2vec_vn.html)*“*, Đắm mình vào học sâu [Online].
3. Nguyễn Đình Mạnh (12/2020), *“Nghiên cứu các phương pháp tính toán độ tương tự của văn bản luật tiếng việt”*, Đại học Quốc gia Hà Nội, Trường Đại học Công Nghệ.
4. Bùi Văn Hợp (July 4, 2021), Giới thiệu về mô hình SVM – Phần 1.
5. Huỳnh Chí Trung (2020), Giới thiệu về Support Vector Machine (SVM).

**Tiếng Anh**

1. Francois Chollet (September 29, 2017), A ten-minute introduction to sequencq-to-sequence learning in Keras
2. NLP Course documentation, Sequence-to-sequence models sequence-to-sequence-models