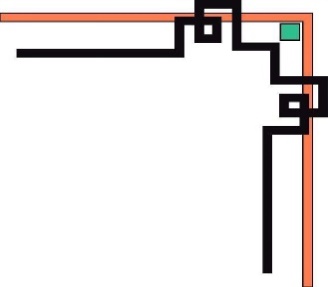
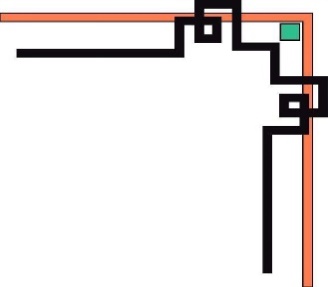
KHOA CÔNG NGHỆ THÔNG TIN



-------------o0o------------

****

**BÁO CÁO BÀI TẬP LỚN**

**XỬ LÝ NGÔN NGỮ TỰ NHIÊN**

**ĐỀ TÀI: PHÂN TÍCH QUAN ĐIỂM**

|  |  |
| --- | --- |
| Sinh viên thực hiện: | Ung Minh Hoài |
| Lớp: | Khoa học máy tính K42 |
| Giảng viên hướng dẫn | TS. Lê Quang Hùng |
|  |  |

Quy Nhơn, tháng 11 năm 2022

**Mục lục**

[**I. Giới thiệu** 4](#_Toc119031641)

[**1. Xử lý ngôn ngữ tự nhiên là gì?** 4](#_Toc119031642)

[**2. Các ứng dụng của xử lý ngôn ngữ tự nhiên** 5](#_Toc119031643)

[**3. Quy trình xử lý ngôn ngữ tự nhiên** 5](#_Toc119031644)

[**II. Cơ sở lý thuyết** 6](#_Toc119031645)

[**1. Thuật toán học máy** 6](#_Toc119031646)

[**1.1. Bag of words** 6](#_Toc119031647)

[**1.2. Decision Tree** 9](#_Toc119031648)

[**1.3. Random Forest** 9](#_Toc119031649)

[**1.4. Logistic Regression** 9](#_Toc119031650)

[**2. Thuật toán học sâu** 10](#_Toc119031651)

[**2.1. Mô hình BERT** 10](#_Toc119031652)

[**2.2. Transformer** 15](#_Toc119031653)

[**3. Các chỉ số đánh giá độ chính xác** 28](#_Toc119031654)

[**3.1. F1-score** 28](#_Toc119031655)

[**3.2. Ma trận nhầm lẫn** 29](#_Toc119031656)

[**III. Phát biểu bài toán** 30](#_Toc119031657)

[**1. Định nghĩa bài toán** 30](#_Toc119031658)

[**2. Cách tiếp cận** 31](#_Toc119031659)

[**3. Chuẩn bị dữ liệu** 31](#_Toc119031660)

[**4. Phân tích dữ liệu** 31](#_Toc119031661)

[**a. Thu thập dữ liệu** 32](#_Toc119031662)

[**b. Phân tích dữ liệu khám phá (EDA)** 32](#_Toc119031663)

[**c. Tiền xử lý dữ liệu** 33](#_Toc119031664)

[**d. Kỹ thuật tính năng** 33](#_Toc119031665)

[**e. Lựa chọn thuật toán** 33](#_Toc119031666)

[**f. Huấn luyện mô hình** 33](#_Toc119031667)

[**IV. Thực nghiệm và kết quả** 34](#_Toc119031668)

[**1. Sử dụng các thuật toán học máy** 34](#_Toc119031669)

[**1.1. Cài đặt các thư viện cần thiết và đọc dữ liệu lên dataframe** 34](#_Toc119031670)

[**1.2. Phân tích dữ liệu khám phá (EDA)** 35](#_Toc119031671)

[**1.3. Tiền xử lý dữ liệu** 38](#_Toc119031672)

[**1.4. Vec-tơ hóa – Mô hình hóa – Đánh giá mô hình** 43](#_Toc119031673)

[**2. Sử dụng deep learning** 51](#_Toc119031674)

[**2.1. Cài đặt các thư viện cần thiết và đọc dữ liệu lên dataframe** 51](#_Toc119031675)

[**2.2. Phân tích dữ liệu khám phá (EDA)** 52](#_Toc119031676)

[**2.3. Tiền xử lý dữ liệu** 54](#_Toc119031677)

[**2.4. Chuẩn bị dữ liệu** 56](#_Toc119031678)

[**2.5. Model Building – Xây dựng mô hình** 57](#_Toc119031679)

[**2.6. Đánh giá mô hình** 61](#_Toc119031680)

[**2.7. So sánh với mô hình sử dụng bộ dữ liệu tiếng Việt** 63](#_Toc119031681)

[**V. Kết luận** 64](#_Toc119031682)

[**1. Những điểm đạt được** 64](#_Toc119031683)

[**2. Những điểm hạn chế** 64](#_Toc119031684)

[**VI. Tài liệu tham khảo** 65](#_Toc119031685)

1. **Giới thiệu**
2. **Xử lý ngôn ngữ tự nhiên là gì?**

Natural Language Processing (Xử lý ngôn ngữ tự nhiên) viết tắt là NLP là một lĩnh vực con của ngôn ngữ học, khoa học máy tính và trí tuệ nhân tạo liên qua đến các tương tác giữa máy tính và ngôn ngữ con người, đặc biệt là cách lập trình máy tính để xử lý và phân tích lượng lớn dữ liệu ngôn ngữ tự nhiên. Mục tiêu là một máy tính có khả năng “hiểu” nội dung của các tài liệu, bao gồm các sắc thái ngữ cảnh của ngôn ngữ bên trong chúng. Sau đó, công nghệ này có thể trích xuất chính xác thông tin và hiểu biết sâu sắc có trong các tài liệu cũng như tự phân loại và sắp xếp các tài liệu.

Trong trí tuệ nhân tạo thì xử lý ngôn ngữ tự nhiên là một trong những phần khó nhất vì nó liên quan đến việc phải hiểu ý nghĩa ngôn ngữ - công cụ hoàn hảo nhất của tư duy và giao tiếp.

Ba cấp độ phân tích ngôn ngữ khác nhau được thực hiện trước khi thực hiện NLP:

* Ngữ nghĩa - quan hệ giữa các từ, câu và đoạn văn (nghĩa của từ)
* Cú pháp - quản trị cấu trúc của văn bản (ngữ pháp)
* Ngữ dụng - cách ngữ cảnh góp phần tạo nên ý nghĩa (hội thoại)

Xử lý ngôn ngữ tự nhiên đề cập đến các khía cạnh khác nhau của ngôn ngữ như:

* Hình thái - cấu trúc và nội dung của các dạng từ (thì)
* Âm vị học - một tổ chức có hệ thống các âm thanh trong ngôn ngữ

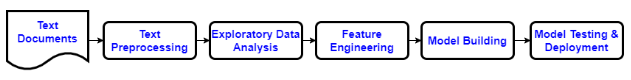
1. **Các ứng dụng của xử lý ngôn ngữ tự nhiên**

Các ứng dụng cơ bản của xử lý ngôn ngữ tự nhiên:

* Chuyển ngữ máy.
* Nhận dạng giọng nói.
* Phân tích cảm xúc.
* Tóm tắt văn bản.
* Chatbot.
* Phân loại văn bản.
* Nhận dạng ký tự.
* Kiểm tra chính tả.
* Phát hiện thư rác.
* Tự động hoàn thành.

1. **Quy trình xử lý ngôn ngữ tự nhiên**

Bất kể định dạng dữ liệu văn bản nào, các bước được sử dụng để giải quyết các vấn đề NLP ít nhiều vẫn giống nhau. Các bước chính cần phải tuân theo khi giải quyết các vấn đề NLP như sau.

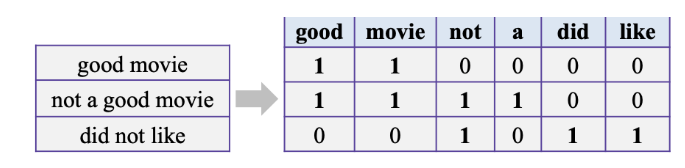


**Hình.** Quy trình xử lý ngôn ngữ tự nhiên

Trong bước tiền xử lý văn bản, chúng ta loại bỏ tất cả sự lộn xộn và nhiễu khỏi văn bản. Sau đó, chúng ta thực hiện phân tích dữ liệu khám phá để hiểu dữ liệu. Dựa trên hiểu biết của chúng ta từ phân tích dữ liệu, chúng ta tạo ra các tính năng mới trong bước kỹ thuật tính năng. Bây giờ một khi chúng ta đã định dạng tốt dữ liệu với các tính năng, để tạo mô hình ML theo yêu cầu của chúng ta. Trong bước cuối cùng, chúng ta kiểm tra mô hình của mình và triển khai nó trong sản xuất.

1. **Cơ sở lý thuyết**
2. **Thuật toán học máy** 
   1. **Bag of words**

Chúng ta không thể cung cấp văn bản (từ) trực tiếp vào các mô hình NLP hoặc ML vì tất cả các thuật toán đều hoạt động trên các con số. Do đó Bag of words - BOG được sử dụng để xử lý trước các văn bản. Ở đây tổng số lần xuất hiện của mỗi từ được tính và được lưu giữ dưới dạng “TÚI TỪ”.:



**Hình.** Ví dụ về túi từ sử dụng CountVectorization

Mô hình Bag-of-words được chia làm 3 loại:

* Count Occurrence (Đếm số lần xuất hiện)
* Normalized Count Occurrence (Đếm số lần xuất hiện được chuẩn hóa)
* TF-IDF
* **Count-vectorizer**

CountVectorizer cung cấp và triển khai cả đếm lần xuất hiện và mã hóa (tokenization). Nó xây dựng một kho từ vựng của những từ đã biết. Nó sử dụng từ vựng này để mã hóa dữ liệu văn bản mới. Mặc dù nó là một giải pháp tốt, có một số hạn chế khi sử dụng CountVectorizer chẳng hạn như các từ không liên quan xảy ra quá nhiều lần

* **TF-IDF**

Term Frequency - inverse document frequency (Tần suất thuật ngữ - tần suất tài liệu nghịch đảo) được định nghĩa là một thống kê số nhằm phản ánh tầm quan trọng của một từ đối với một tài liệu trong một bộ sưu tập / kho tài liệu

* TF: Nó là thước đo tần suất một thuật ngữ (t) xuất hiện trong tài liệu:

**tf = n / số thuật ngữ trong tài liệu**

* IDF: là thước đo mức độ quan trọng của một thuật ngữ

**idf = log (số tài liệu / số tài liệu có thuật ngữ 't')**

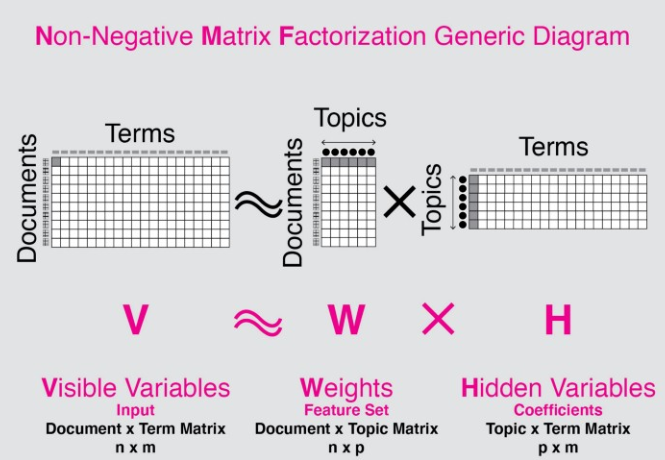
* **TF-IDF x NFM**

1. **Định nghĩa**

NMF là một dạng Lập mô hình chủ đề (Topic Modellings) - nghệ thuật trích xuất các chủ đề có ý nghĩa được lặp lại thông qua một kho tài liệu. Một kho ngữ liệu bao gồm một tập hợp các chủ đề được nhúng trong tài liệu của nó. Một tài liệu bao gồm một hệ thống phân cấp các chủ đề. Một chủ đề bao gồm một hệ thống phân cấp các thuật ngữ.

1. **Kiến trúc của NMF**

NMF là viết tắt của Phân tích ngữ nghĩa tiềm ẩn với phương pháp 'Dữ liệu hóa ma trận không âm' (Non-negative Matrix-Factorization) là một dạng của Mô hình chủ đề được sử dụng để phân tách ma trận thuật ngữ tài liệu thành hai ma trận nhỏ hơn - ma trận chủ đề tài liệu (U) và ma trận chủ đề thuật ngữ (W) - mỗi ma trận được điền với các xác suất không chuẩn hóa.



**Hình.** Ma trận thuật ngữ tài liệu

Các giá trị trong V có thể được tính gần đúng thông qua phép nhân ma trận của W và H. Ví dụ, hộp màu xám trong V có thể được tìm thấy thông qua phép nhân ma trận của hàng đầu tiên của W với cột đầu tiên của H, cũng được tô màu xám ở hàng đầu tiên của V có thể được tính gần đúng thông qua phép nhân ma trận của hàng đầu tiên của W với toàn bộ ma trận của H.

* **V là Visible**

Trong ma trận V , mỗi hàng đại diện cho một Tài liệu bao gồm tần suất các Cụm từ hay còn gọi là số lần xuất hiện của mỗi từ. Mỗi cột đại diện cho các biến hiển thị .

* **W là trọng lượng**

Trong ma trận W , mỗi hàng đại diện cho một Tài liệu bao gồm các xác suất không chuẩn hóa của các Chủ đề. Mỗi cột đại diện cho một đặc điểm ngữ nghĩa lặp lại trong toàn bộ kho ngữ liệu.

* **H là ẩn**

Trong ma trận H, mỗi hàng đại diện cho một Chủ đề hoặc đặc điểm ngữ nghĩa bao gồm các tần số thuật ngữ. Mỗi cột đại diện cho một biến hiển thị. Hai thuật ngữ xuất hiện thường xuyên cùng nhau tạo thành một chủ đề và mỗi thuật ngữ mang lại nhiều ý nghĩa theo ngữ cảnh hơn cho thuật ngữ mà nó được nhóm lại với nhau.Nếu một thuật ngữ xuất hiện thường xuyên trong hai chủ đề, thì những chủ đề đó có thể có liên quan.

* **Tái tạo V**

Ma trận W và H có thể được sử dụng để tái tạo lại gần đúng V thông qua phép nhân ma trận.

1. **Cách sử dụng NMF**

NMF (Phân tích nhân tử ma trận không âm) là một kỹ thuật rất hiệu quả để chọn thông tin quan trọng nhất có trong kho tài liệu đào tạo, nó là một dạng mô hình hóa Chủ đề và có thể được sử dụng trong nhiều lĩnh vực như Hệ thống khuyến nghị, Mô hình xếp hạng, Phân loại chủ đề…

Cách sử dụng NMF như sau:

* Chuyển đổi văn bản thô thành Vectơ bằng cách sử dụng vectơ TF-IDF
* Chuyển các vector thành dạng NFM, tách ma trận NXM thành NXP và PXM tương ứng,
* Trong đó N là tổng số câu, P là số chủ đề, M là số hạng trong câu.
* Sau đó, dựa trên kết quả ma trận NMF, tôi đã tạo một khung dữ liệu với văn bản và nhãn là các thuộc tính.
* Số lượng chủ đề và n\_iter là siêu tham số, có thể điều chỉnh chúng cho phù hợp bằng cách sử dụng GridSearch hoặc RandomizedSearch
  1. **Decision Tree**

Decission Trees (Cây quyết định) là cây mà mỗi nút đại diện cho một đặc điểm (thuộc tính), mỗi liên kết (nhánh) đại diện cho một quyết định (quy tắc) và mỗi lá đại diện cho một kết quả (giá trị phân loại hoặc tiếp tục).

* 1. **Random Forest**

Random Forest (Rừng ngẫu nhiên) là một thuật toán học có giám sát. “Forest” mà nó xây dựng là một tập hợp các cây quyết định, thường được đào tạo theo phương pháp “bagging”. Ý tương chung của phương pháp “bagging” là sự kết hợp của các mô hình học tập sẽ làm tăng kết quả chung.

Nói một cách đơn giản: rừng ngẫu nhiên xây dựng nhiều cây quyết định và kết hợp chúng lại với nhau để có được dự đoán chính xác và ổn định hơn.

Một lợi thế lớn của Random Forest là nó có thể được sử dụng cho cả các bài toán phân loại và hồi quy.

* 1. **Logistic Regression**

Logistic Regression (Hồi quy logistic) là một thuật toán phân loại được sử dụng để gán các quan sát cho một tập hợp các lớp rời rạc. Một số ví dụ về vấn đề phân loại Email spam hoặc không phải spam, giao dịch trực tuyến gian lận hoặc không gian lận, khối u ác tính hoặc lành tính. Hồi quy logistic biến đổi đầu ra của nó bằng cách sử dụng hàm sigmoid logistic để trả về giá trị xác suất.

Các loại hồi quy logistic:

* Binary
* Multi
* Ordinal

1. **Thuật toán học sâu**
   1. **Mô hình BERT**
2. **Tổng quan**

BERT - Bidirectional Encoder Representations from Transformers, do Google phát triển được tạo và xuất bản vào năm 2018 bởi Jacob Devlin và các đồng nghiệp. Nó là một khuôn khổ học máy mã nguồn mở để xử lý ngôn ngữ tự nhiên (NLP). BERT được thiết kế để giúp máy tính hiểu ý nghĩa của ngôn ngữ mơ hồ trong văn bản bằng cách sử dụng văn bản xung quanh để thiết lập ngữ cảnh. Khung BERT đã được đào tạo trước bằng cách sử dụng văn bản từ Wikipedia và BooksCorpus và có thể được tinh chỉnh theo từng nhiệm vụ cụ thể. Google đang tận dụng BERT để hiểu rõ hơn các tìm kiếm của người dùng.

Không quá lời khi nói rằng BERT đã thay đổi đáng kể bối cảnh NLP. Hãy tưởng tượng sử dụng một mô hình duy nhất được đào tạo trên một tập dữ liệu lớn không có nhãn để đạt được kết quả hiệu quả trên 11 nhiệm vụ NLP riêng lẻ. Và tất cả những điều này với một chút tinh chỉnh.

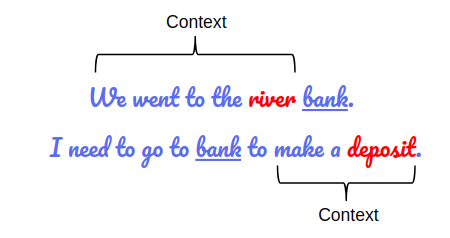
BERT đã truyền cảm hứng cho nhiều kiến ​​trúc NLP, phương pháp đào tạo và mô hình ngôn ngữ gần đây, chẳng hạn như Google’s TransformerXL, OpenAI’s GPT-2, XLNet, ERNIE2.0, RoBERTa, v.v.

1. **BERT là gì?**

BERT là một mô hình học sâu trong đó mọi yếu tố đầu ra được kết nối với mọi yếu tố đầu vào và trọng số giữa chúng được tính toán động dựa trên kết nối của chúng. Trong NLP, quá trình này được gọi là sự chú ý (self-attention). Một số điểm cần chú ý:

* Đầu tiên, BERT dựa trên kiến trúc Transformer.
* Thứ hai, BERT được đào tạo trước trên một kho ngữ liệu lớn của văn bản không có nhãn bao gồm toàn bộ Wikipedia (tức là 2.500 triệu từ!) và Book Corpus (800 triệu từ).
* Thứ ba, BERT là một mô hình hai chiều sâu sắc. Hai chiều có nghĩa là BERT học thông tin từ cả bên trái và bên phải của ngữ cảnh mã thông báo trong giai đoạn đào tạo.

Có thể có hai câu có cùng một từ nhưng nghĩa của chúng có thể hoàn toàn khác nhau dựa trên những gì đứng trước hoặc sau như chúng ta có thể thấy ở đây dưới đây.



**Hình.** Ví dụ minh họa cùng một từ nhưng khác nhau về ngữ nghĩa

Nếu không xem xét các ngữ cảnh, máy sẽ không thể thực sự hiểu được ý nghĩa và nó có thể đưa ra các phản hồi rác hết lần này đến lần khác, điều này không thực sự là một điều tốt. Nhưng BERT đã khắc phục điều này.

* Thứ tư, lợi thế lớn nhất của BERT là nó mang lại chuyển động ImageNet với nó và khía cạnh ấn tượng nhất của BERT là chúng tôi có thể tinh chỉnh nó bằng cách chỉ thêm một vài lớp đầu ra bổ sung để tạo ra các mô hình cho nhiều loại nhiệm vụ NLP.

1. **Ứng dụng của BERT**

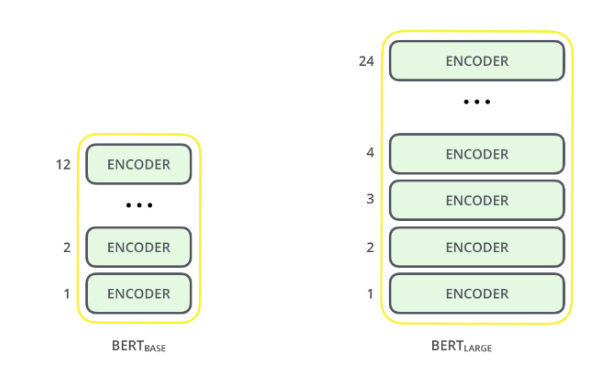
BERT hiện đang được sử dụng tại Google để tối ưu hóa việc giải thích các truy vấn tìm kiếm của người dùng. BERT vượt trội ở một số chức năng giúp điều này trở nên khả thi, bao gồm:

* Các tác vụ tạo ngôn ngữ dựa trên trình tự đến trình tự như:
* Câu trả lời câu hỏi
* Tóm tắt trừu tượng
* Dự đoán câu
* Tạo phản hồi hội thoại
* Các nhiệm vụ hiểu ngôn ngữ tự nhiên như:
* Polysemy và Coreference (các từ có âm thanh hoặc trông giống nhau nhưng có ý nghĩa khác nhau)
* Sự phân biệt từ ngữ
* Suy luận ngôn ngữ tự nhiên
* Phân loại tình cảm

1. **Kiến trúc của BERT**

* BERT là một bộ mã hóa nhiều lớp. Có hai mô hình chính:
* BERT cơ sở - 12 lớp (khối biến áp), 12 đầu chú ý và 110 triệu thông số.
* BERT Large - 24 lớp, 16 đầu chú ý và 340 triệu thông số.

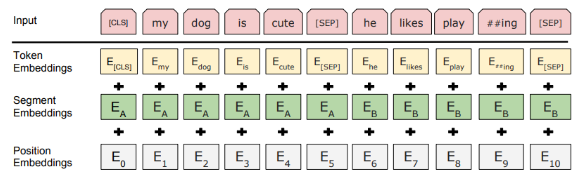
Đầu vào phải được cung cấp cho bộ mã hóa đầu tiên



**Hình 5.** Kiến trúc của BERT

Bộ mã hóa BERT mong đợi một chuỗi các mã thông báo. Hình ảnh dưới đây cho thấy cách các mã thông báo được xử lý và chuyển đổi.

* [CLS] là một mã thông báo đặc biệt được chèn vào đầu câu đầu tiên. [SEP] được chèn vào cuối mỗi câu.
* Tạo nhúng phân đoạn (segment embeddings) bằng cách thêm phân đoạn 'A' hoặc 'B' để phân biệt giữa các câu.
* Nhúng vị trí (positional embedding) cũng được thêm vào mỗi mã thông báo để chỉ ra vị trí của nó trong chuỗi.



**Hình 6.** Biểu diễn bộ quy tắc đại diện cho các ngôn ngữ

Tổng của ba lần nhúng trên là đầu vào cuối cùng cho bộ mã hóa BERT.

1. **BERT đào tạo trước**

* Làm thế nào để BERT đạt được đào tạo hai chiều? Nó sử dụng hai phương pháp:

1. **Masked Language Model - Mô hình ngôn ngữ được che giấu**

Trong trình tự, chúng ta che dấu ngẫu nhiên một số phần trăm từ, bằng cách thay thế chúng bằng mã thông báo [MASK]. Che đi 15% các từ đầu vào. Nó được huấn luyện để dự đoán những từ bị che này bằng cách sử dụng ngữ cảnh của những từ còn lại

* **Ví dụ:** "Tôi thích đi xe đạp trong mùa xuân" -> Tôi thích đi xe đạp trong mùa [MASK].

Một vấn đề ở đây là các mô hình được đào tạo trước sẽ có 15% mã thông báo được che giấu, nhưng khi chúng ta tinh chỉnh các mô hình được đào tạo trước và chuyển đầu vào, chúng ta sẽ không vượt qua các mã thông báo bị che. Để giải quyết vấn đề này, trong số 15% mã thông báo được chọn để che: 80% - thực sự được thay thế bằng mã thông báo [MASK], 10% thời gian mã thông báo được thay thế bằng mã thông báo ngẫu nhiên và phần còn lại không thay đổi

**2. Next Sentence Prediction - Dự đoán câu tiếp theo.**

Để hiểu mối quan hệ giữa hai câu, BERT sử dụng đào tạo NSP. Mô hình nhận được các cặp câu làm đầu vào và nó được huấn luyện để dự đoán xem câu thứ hai có phải là câu tiếp theo với câu đầu tiên hay không.

Tóm lại, mục tiêu của Masked LM là tìm ra từ bị che là gì. Mặt khác, dự đoán câu tiếp theo dự đoán nếu câu tiếp theo được kết nối với câu trước, tức là liệu câu thứ hai có sau câu đầu tiên hay không.

1. **Cách tinh chỉnh cho BERT**

BERT có thể được sử dụng cho nhiều tác vụ ngôn ngữ khác nhau, trong khi chỉ thêm một lớp nhỏ vào mô hình cốt lõi

* **Sequence Classification Tasks - Nhiệm vụ phân loại trình tự**

Các tác vụ phân loại, chẳng hạn như phân tích tình cảm được thực hiện tương tự như phân loại Câu tiếp theo, bằng cách thêm một lớp phân loại lên trên đầu ra của Transformer cho mã thông báo [CLS].

* **Question-Answering Tasks - Nhiệm vụ trả lời câu hỏi**

Trong các tác vụ Trả lời Câu hỏi (ví dụ: SQuAD v1.1), phần mềm nhận được một câu hỏi liên quan đến một chuỗi văn bản và được yêu cầu đánh dấu câu trả lời trong chuỗi. Sử dụng BERT, mô hình Hỏi & Đáp có thể được đào tạo bằng cách học thêm hai vectơ đánh dấu phần đầu và phần cuối của câu trả lời.

* **Single Sentence Tagging Tasks - Nhiệm vụ gắn thẻ câu đơn**

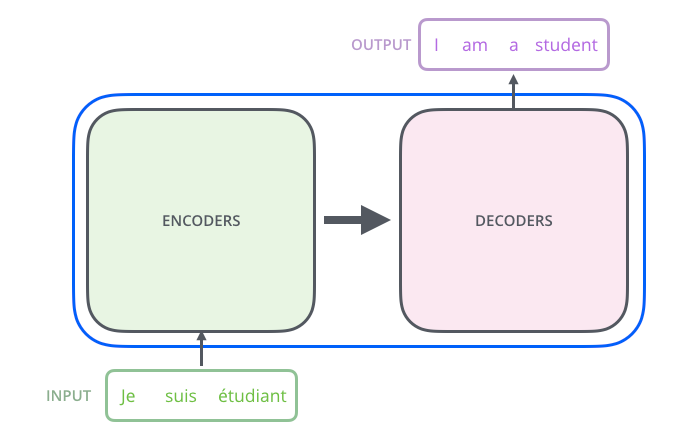
Trong Nhận dạng thực thể được đặt tên (NER), phần mềm nhận một chuỗi văn bản và được yêu cầu đánh dấu các loại thực thể khác nhau (Người, Tổ chức, Ngày, v.v.) xuất hiện trong văn bản. Sử dụng BERT, một mô hình NER có thể được đào tạo bằng cách đưa vectơ đầu ra của mỗi mã thông báo vào một lớp phân loại dự đoán nhãn NER.

* 1. **Transformer**
     1. **Các thành phần chính của mô hình**

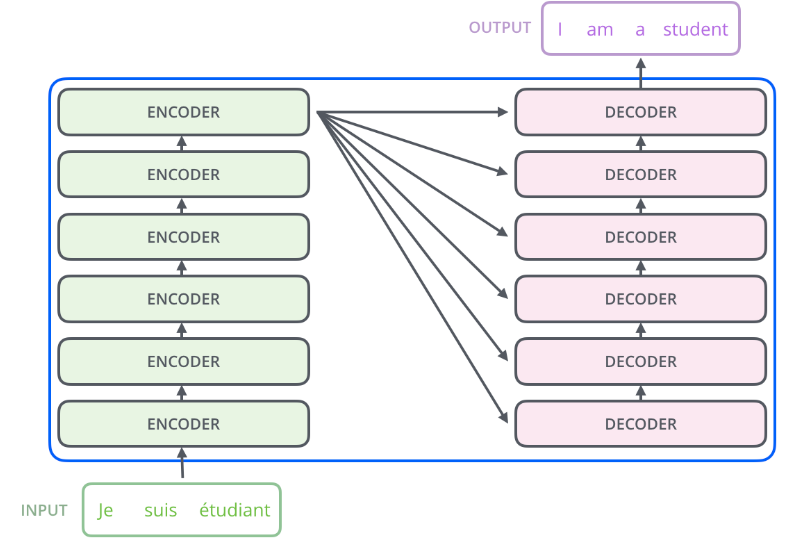
Thông qua một ứng dụng dịch máy để hiểu cụ thể hoạt động của Transformer như thế nào. Nó sẽ lấy một câu bằng một ngôn ngữ và xuất ra bản dịch của nó bằng một ngôn ngữ khác.



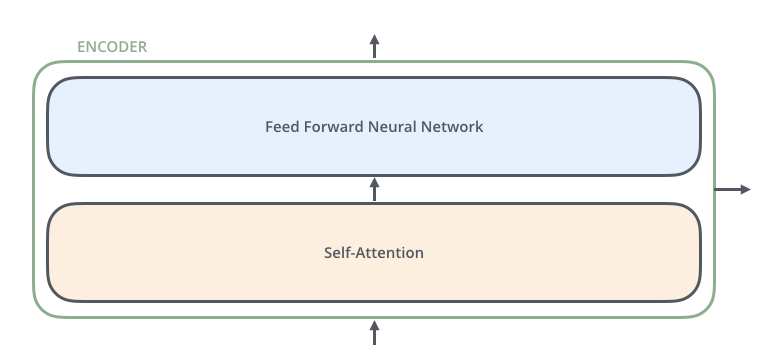
**Tiếp theo,** chúng ta thấy một thành phần mã hóa (encoders), một thành phần giải mã (decoders) và các kết nối giữa chúng.



Thành phần mã hóa là một chồng các bộ mã hóa. Thành phần giải mã là một chồng các bộ giải mã của cùng một số.

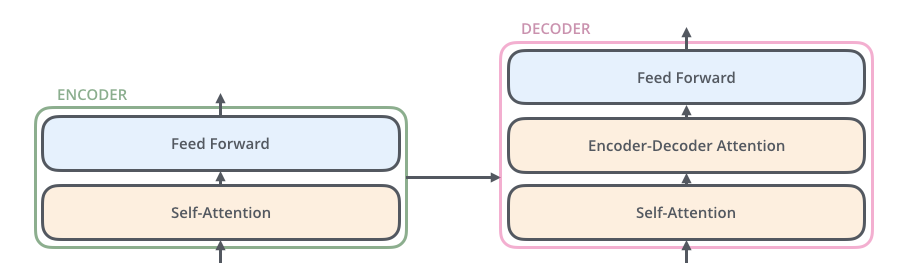


Các bộ mã hóa đều giống nhau về cấu trúc (nhưng chúng không có chung trọng số). Mỗi cái được chia thành hai lớp con:



* Đầu vào của bộ mã hóa trước tiên đi qua một lớp tự chú ý (Self-Attention) - một lớp giúp bộ mã hóa xem xét các từ khác trong câu đầu vào khi nó mã hóa một từ cụ thể.
* Các đầu ra của lớp tự chú ý được đưa đến một mạng nơ-ron chuyển tiếp (Feed Forward Neural Network. Mạng chuyển tiếp giống hệt nhau được áp dụng độc lập cho từng vị trí.

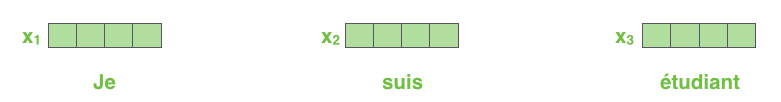
Bộ giải mã có cả hai lớp đó, nhưng giữa chúng là một lớp chú ý giúp bộ giải mã tập trung vào các phần có liên quan của câu đầu vào.



* + 1. **Cách luân chuyển giữa các thành phần**

Bây giờ chúng ta đã thấy các thành phần chính của mô hình, hãy bắt đầu xem xét các vectơ / tensor khác nhau và cách chúng luân chuyển giữa các thành phần này để biến đầu vào của một mô hình được đào tạo thành đầu ra.

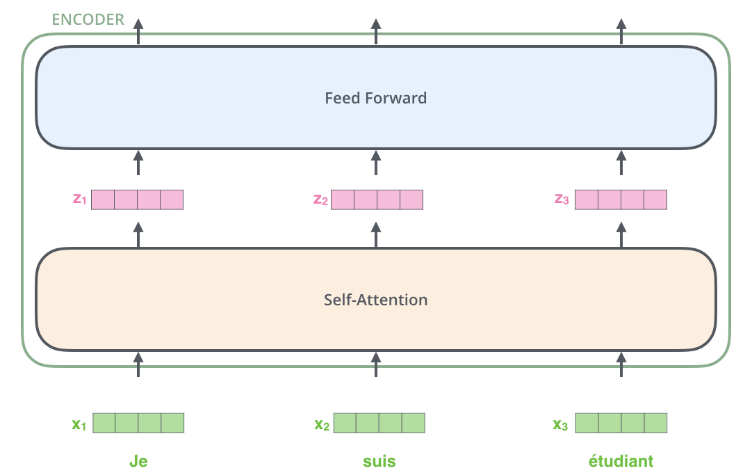
Như trường hợp của các ứng dụng NLP nói chung, chúng ta bắt đầu bằng cách chuyển mỗi từ đầu vào thành một vectơ bằng cách sử dụng một thuật toán Embedding.



Mỗi từ được nhúng vào một vectơ có kích thước 512. Chúng ta sẽ biểu diễn các vectơ đó bằng các hộp đơn giản.

Việc nhúng chỉ xảy ra trong bộ mã hóa ở dưới cùng. Điểm trừu tượng chung cho tất cả các bộ mã hóa là chúng nhận được danh sách các vectơ, mỗi vectơ có kích thước 512 - Trong bộ mã hóa dưới cùng sẽ là từ nhúng, nhưng trong các bộ mã hóa khác, nó sẽ là đầu ra của bộ mã hóa nằm ngay bên dưới . Kích thước của danh sách này là siêu tham số mà chúng ta có thể đặt - về cơ bản nó sẽ là độ dài của câu dài nhất trong tập dữ liệu đào tạo của chúng ta.

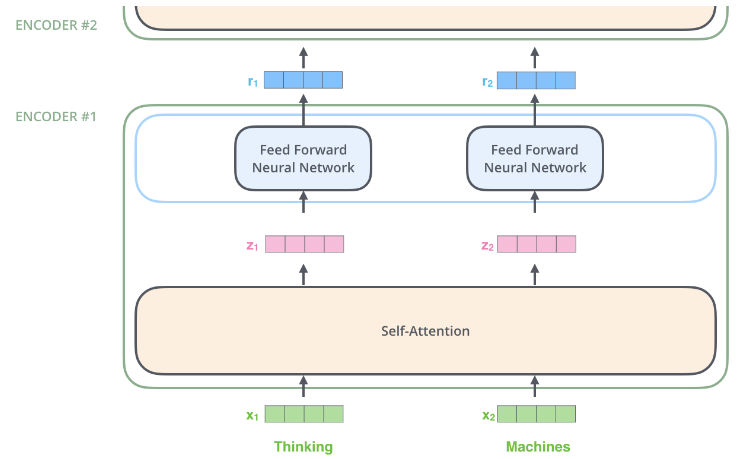
Sau khi nhúng các từ vào trình tự đầu vào, mỗi từ trong số chúng sẽ đi qua từng lớp trong số hai lớp của bộ mã hóa.



Ở đây chúng ta bắt đầu thấy một thuộc tính chính của Transformer, đó là từ ở mỗi vị trí đi qua đường dẫn riêng của nó trong bộ mã hóa. Có sự phụ thuộc giữa các đường dẫn này trong lớp tự chú ý. Tuy nhiên, lớp chuyển tiếp không có những phụ thuộc đó và do đó, các đường dẫn khác nhau có thể được thực hiện song song trong khi chảy qua lớp chuyển tiếp.

**Tiếp theo,** chúng tôi sẽ chuyển ví dụ sang một câu ngắn hơn và chúng tôi sẽ xem xét điều gì xảy ra trong mỗi lớp con của bộ mã hóa.

Như đã đề cập, một bộ mã hóa nhận một danh sách các vectơ làm đầu vào. Nó xử lý danh sách này bằng cách chuyển các vectơ này vào một lớp 'tự chú ý', sau đó vào mạng nơ-ron chuyển tiếp, sau đó gửi đầu ra lên bộ mã hóa tiếp theo.



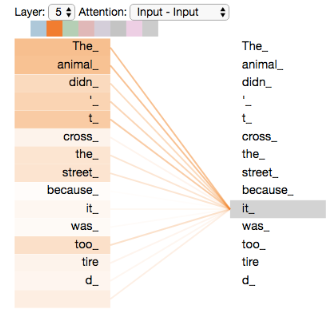
* + 1. **Self-Attention – Tự chú ý**
* Giả sử câu sau là câu đầu vào mà chúng tôi muốn dịch:

“**The animal didn't cross the street because it was too tired**”

“it” trong câu này ám chỉ điều gì? Nó đề cập đến street hay animal? Đó là một câu hỏi đơn giản đối với con người, nhưng không đơn giản đối với một thuật toán. Khi mô hình đang xử lý từ “it”, sự tự chú ý cho phép nó liên kết “it” với “animal”.

Khi mô hình xử lý từng từ (từng vị trí trong chuỗi đầu vào), sự chú ý của bản thân cho phép nó xem xét các vị trí khác trong chuỗi đầu vào để tìm manh mối có thể giúp dẫn đến mã hóa tốt hơn cho từ này.

Nếu đã quen thuộc với RNN, hãy nghĩ cách duy trì trạng thái ẩn cho phép RNN kết hợp biểu diễn của nó với các từ / vectơ trước đó mà nó đã xử lý với trạng thái hiện tại mà nó đang xử lý. Tự chú ý là phương pháp mà Transformer sử dụng để đưa "sự hiểu biết" của các từ có liên quan khác vào từ mà chúng tôi hiện đang xử lý.

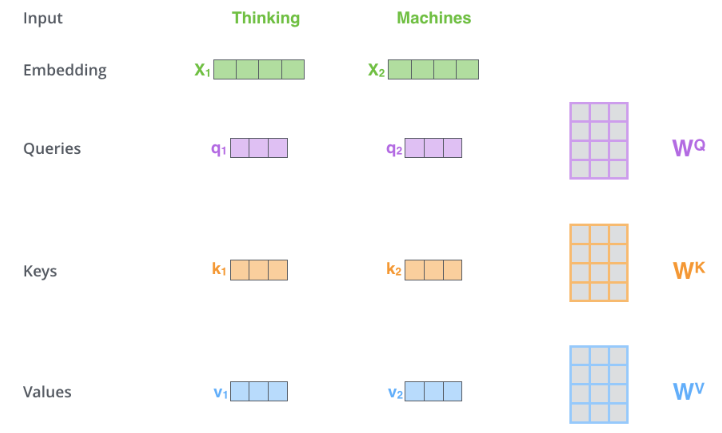


* + 1. **Chi tiết về tự chú ý**

Trước tiên, xem cách tính toán sự chú ý của bản thân bằng cách sử dụng vectơ, sau đó tiến hành xem xét cách nó thực sự được triển khai - bằng cách sử dụng ma trận.

**Bước đầu tiên** trong việc tính toán sự chú ý của bản thân là tạo ba vectơ từ mỗi vectơ đầu vào của bộ mã hóa (trong trường hợp này là nhúng mỗi từ). Vì vậy, đối với mỗi từ, chúng tôi tạo một vectơ Truy vấn (Queries), một vectơ Khóa (Keys) và một vectơ Giá trị (Values). Các vectơ này được tạo ra bằng cách nhân phép nhúng với ba ma trận mà chúng tôi đã đào tạo trong quá trình đào tạo.

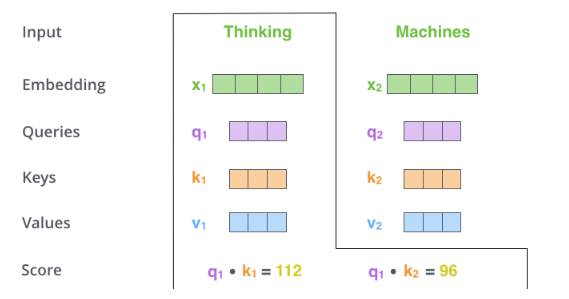
Lưu ý rằng các vectơ mới này có kích thước nhỏ hơn so với vectơ nhúng. Kích thước của chúng là 64, trong khi các vectơ đầu vào / đầu ra nhúng và bộ mã hóa có kích thước là 512. Chúng không phải nhỏ hơn, đây là một lựa chọn kiến ​​trúc để làm cho việc tính toán sự chú ý nhiều nhóm (hầu hết) không đổi.



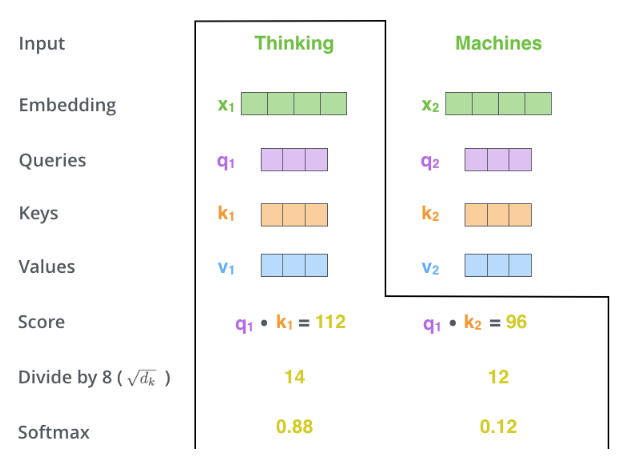
Nhân x1 với ma trận trọng số WQ tạo ra q1 , vectơ "truy vấn" được liên kết với từ đó. Tương tự cho các trường hợp còn lại.

**Bước thứ hai** trong việc tính toán sự chú ý của bản thân là tính điểm. Giả sử chúng ta đang tính toán sự chú ý của bản thân cho từ đầu tiên trong ví dụ này, "Thinking". Chúng ta cần chấm điểm từng từ của câu đầu vào so với từ này. Điểm số xác định mức độ tập trung vào các phần khác của câu đầu vào khi chúng ta mã hóa một từ ở một vị trí nhất định.

Điểm được tính bằng cách lấy tích số chấm của vectơ truy vấn với vectơ chính của từ tương ứng mà chúng ta đang cho điểm. Vì vậy, nếu chúng ta đang xử lý sự tự chú ý cho từ ở vị trí số 1 , điểm đầu tiên sẽ là tích số chấm của q1 và k1 . Điểm thứ hai sẽ là tích số chấm của q1 và k2 .



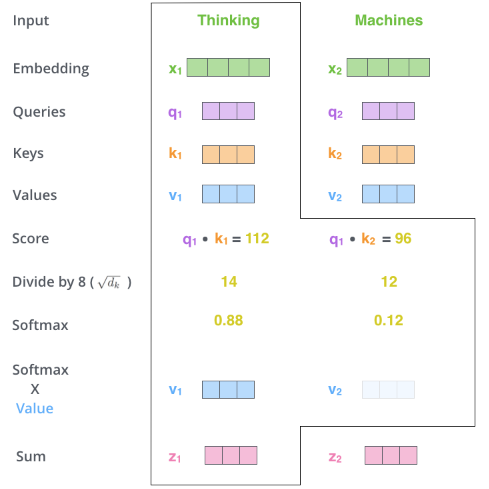
**Bước thứ ba và thứ tư** là chia điểm số cho 8 (căn bậc hai của kích thước của các vectơ. Điều này dẫn đến việc có các gradient ổn định hơn. Có thể có các giá trị khả thi khác ở đây, nhưng đây là default), sau đó chuyển kết quả thông qua hoạt động softmax. Softmax chuẩn hóa các điểm số để tất cả chúng đều dương và cộng đến 1.



Điểm softmax này xác định mức độ mà mỗi từ sẽ được diễn đạt tại vị trí này. Rõ ràng là từ ở vị trí này sẽ có điểm softmax cao nhất, nhưng đôi khi sẽ hữu ích khi xem một từ khác có liên quan đến từ hiện tại.

**Bước thứ năm** là nhân mỗi vectơ giá trị với điểm softmax (để chuẩn bị tính tổng chúng). Trực giác ở đây là giữ nguyên vẹn các giá trị của (các) từ mà chúng ta muốn tập trung vào và loại bỏ các từ không liên quan (bằng cách nhân chúng với các số nhỏ như 0,001 chẳng hạn).

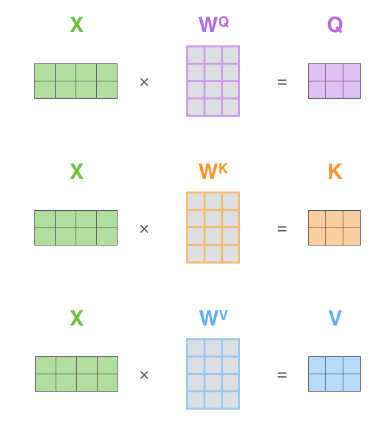
**Bước thứ sáu** là tính tổng các vectơ giá trị có trọng số. Điều này tạo ra đầu ra của lớp tự chú ý tại vị trí này (đối với từ đầu tiên).



Điều đó kết thúc việc tính toán sự chú ý của bản thân. Vectơ kết quả là một vectơ mà chúng ta có thể gửi cùng với mạng nơ-ron chuyển tiếp. Tuy nhiên, trong thực tế triển khai, phép tính này được thực hiện dưới dạng ma trận để xử lý nhanh hơn.

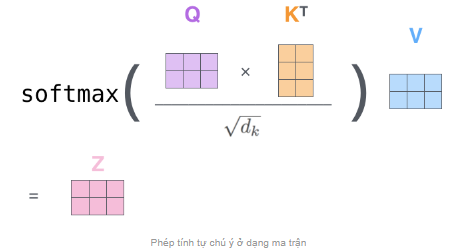
* + 1. **Ma trận tính toán sự chú ý của bản thân**

**Bước đầu tiên** là tính toán các ma trận Truy vấn, Khóa và Giá trị. Chúng ta làm điều đó bằng cách đóng gói các nhúng của chúng ta vào một ma trận X và nhân nó với các ma trận trọng số đã đào tạo ( WQ , WK , WV ).

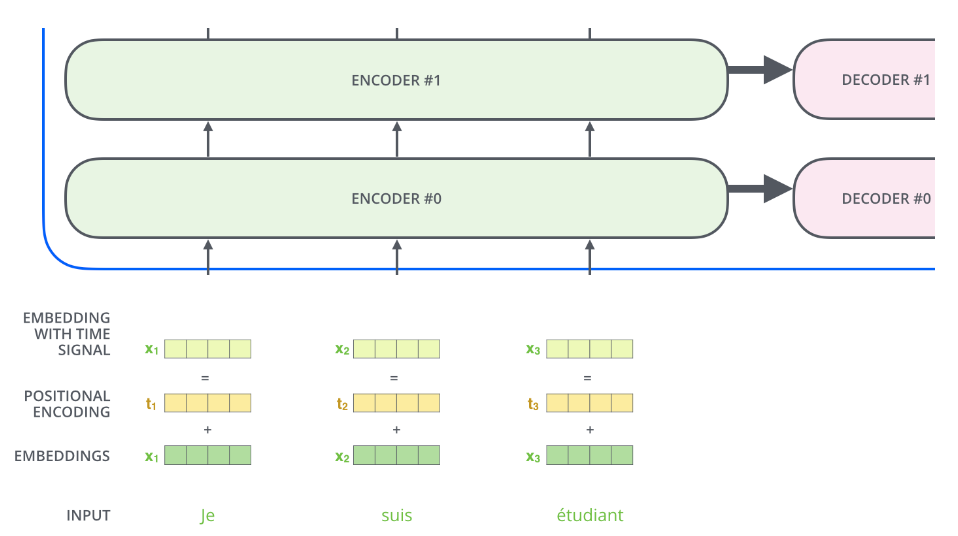


Mỗi hàng trong ma trận X tương ứng với một từ trong câu đầu vào.

**Cuối cùng** , vì chúng ta đang xử lý ma trận, chúng ta có thể cô đọng các bước từ hai đến sáu trong một công thức để tính toán kết quả đầu ra của lớp tự chú ý.

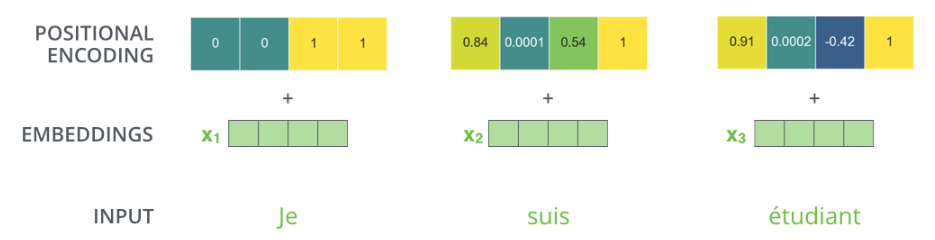


* + 1. **Biểu diễn thứ tự của trình tự bằng cách sử dụng mã hóa vị trí**



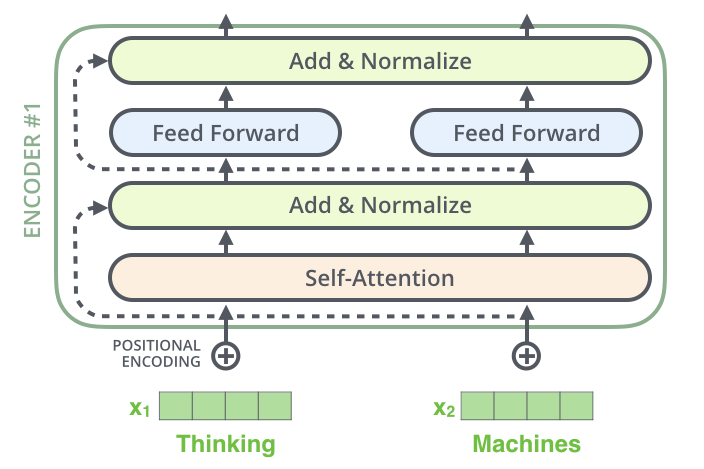
Để cung cấp cho mô hình cảm giác về thứ tự của các từ, chúng ta thêm các vectơ mã hóa vị trí - các giá trị của chúng tuân theo một mẫu cụ thể.

Nếu chúng ta giả định rằng phép nhúng có kích thước là 4, thì các mã hóa vị trí thực tế sẽ giống như sau:

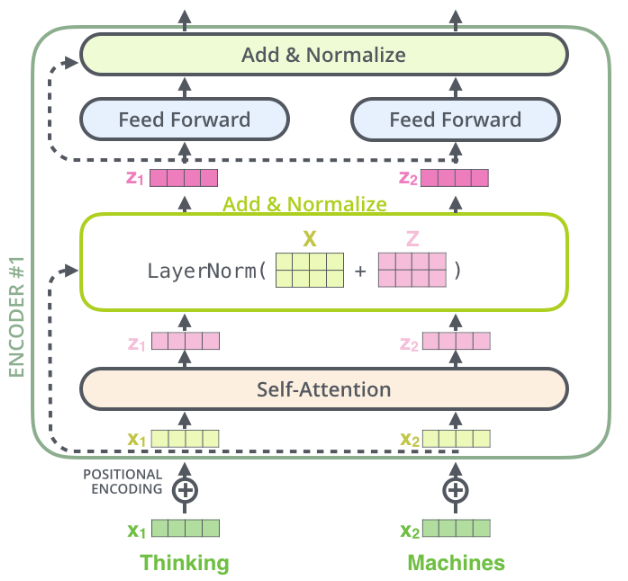


* + 1. **Phần dư**

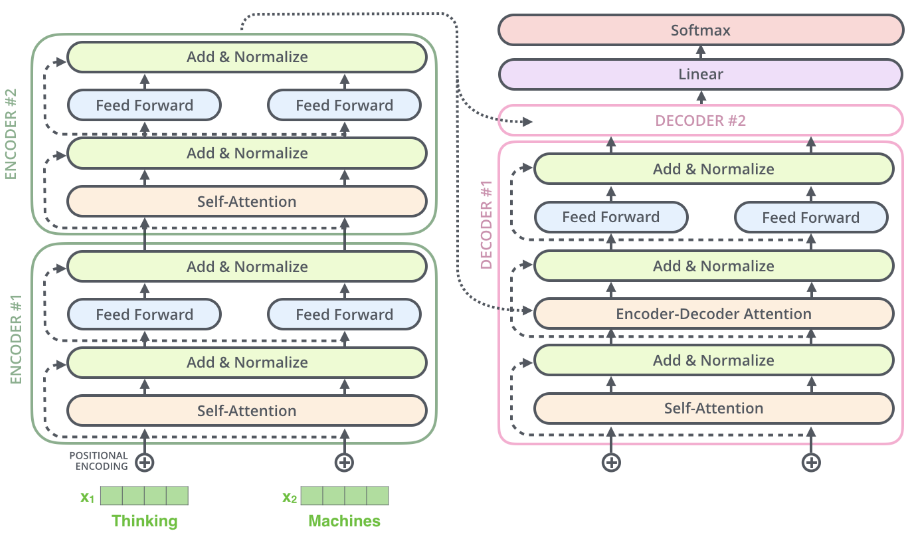
Một chi tiết trong kiến ​​trúc của bộ mã hóa mà chúng ta cần đề cập trước khi tiếp tục, đó là mỗi lớp con (tự chú ý, ffnn) trong mỗi bộ mã hóa có một kết nối dư xung quanh nó và được theo sau bởi bước chuẩn hóa lớp .



Nếu chúng ta hình dung các vectơ và thao tác chuẩn lớp liên quan đến sự chú ý của bản thân, nó sẽ giống như sau:



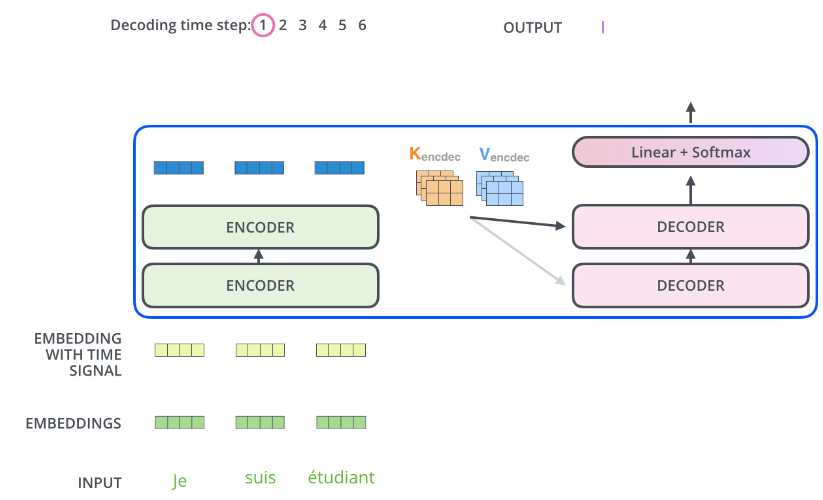
Điều này cũng áp dụng cho các lớp con của bộ giải mã. Nếu chúng ta nghĩ về một Transformer gồm 2 bộ mã hóa và giải mã xếp chồng lên nhau, nó sẽ giống như sau:



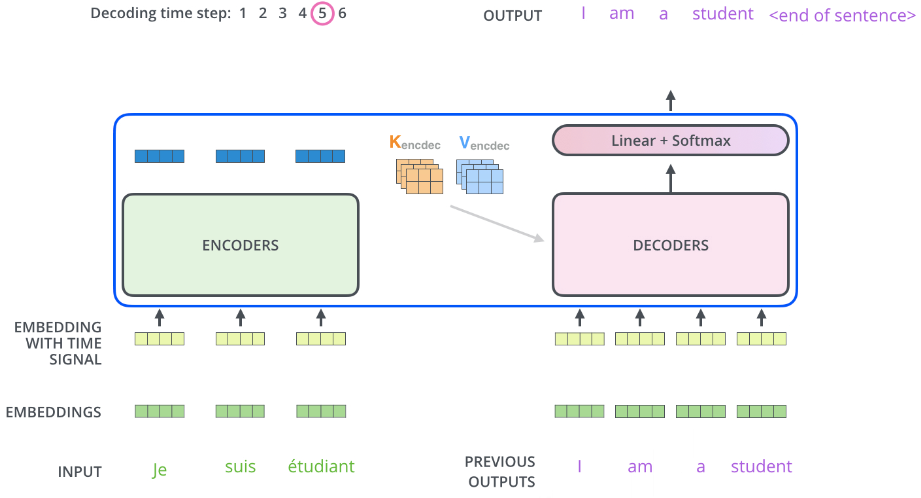
* + 1. **Giải mã**

Đến đây thì cũng đã trình bày hầu hết các khái niệm về phía bộ mã hóa, về cơ bản chúng ta cũng biết cách các thành phần của bộ giải mã hoạt động. Và hãy cùng xem cách chúng hoạt động cùng nhau như thế nào.

Bộ mã hóa bắt đầu bằng cách xử lý chuỗi đầu vào. Đầu ra của bộ mã hóa hàng đầu sau đó được chuyển đổi thành một tập hợp các vectơ chú ý K và V. Các vectơ này sẽ được sử dụng bởi mỗi bộ giải mã trong lớp “encoder-decoder attention” của nó, giúp bộ giải mã tập trung vào những vị trí thích hợp trong chuỗi đầu vào:



* Các bước sau đây lặp lại quy trình cho đến khi hoàn thành đầu ra của nó.



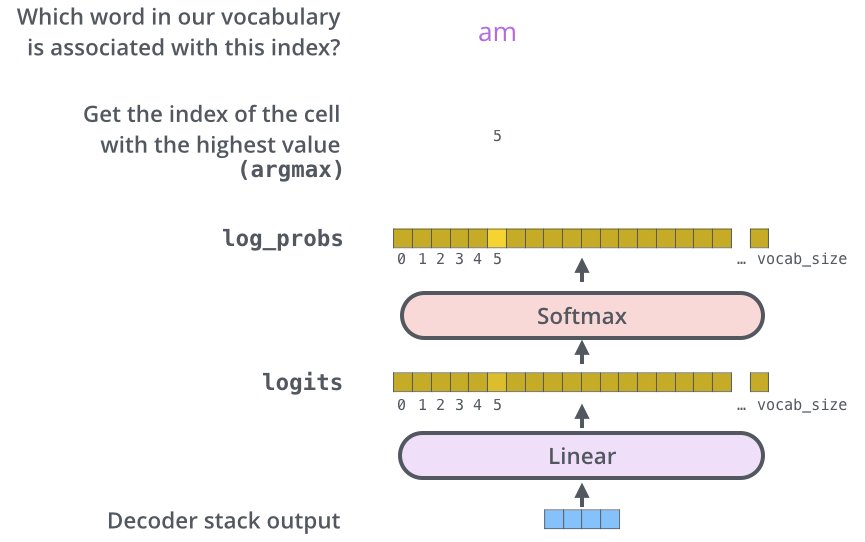
* + 1. **Lớp tuyến tính cuối cùng và lớp Softmax**

Ngăn xếp bộ giải mã xuất ra một vectơ gồm các phao. Làm thế nào để chúng ta chuyển nó thành một từ? Đó là công việc của lớp Tuyến tính cuối cùng, tiếp theo là Lớp Softmax.

Lớp Tuyến tính là một mạng nơ-ron được kết nối đầy đủ đơn giản, chiếu vectơ được tạo ra bởi chồng bộ giải mã, thành một vectơ lớn hơn rất nhiều được gọi là vectơ logits.

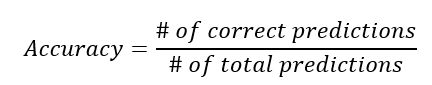
Giả sử rằng mô hình của chúng tôi biết 10.000 từ tiếng Anh duy nhất (“từ vựng đầu ra” của mô hình của chúng tôi) mà nó học được từ tập dữ liệu đào tạo của nó. Điều này sẽ làm cho vectơ logits rộng 10.000 ô - mỗi ô tương ứng với điểm của một từ duy nhất. Đó là cách chúng ta diễn giải đầu ra của mô hình theo sau bởi lớp Tuyến tính.

Sau đó, lớp softmax biến những điểm số đó thành xác suất (tất cả đều dương, tất cả cộng lại là 1,0). Ô có xác suất cao nhất được chọn và từ liên kết với nó được tạo ra làm đầu ra cho bước thời gian này.

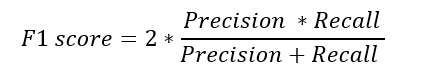


1. **Các chỉ số đánh giá độ chính xác**

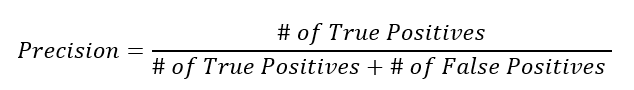
Accuracy (độ chính xác) là số liệu cho các mô hình phân loại đo lường số lượng dự đoán đúng theo tỷ lệ phần trăm của tổng số dự đoán được thực hiện.



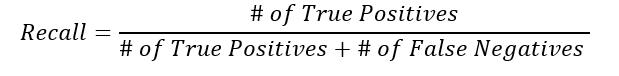
* 1. **F1-score**
* Điểm F1 được định nghĩa là giá trị trung bình hài hòa của **Precision** và **Recall**.



* Trong đó:
* **Precision** có nghĩa là phần trăm kết quả của bạn có liên quan.

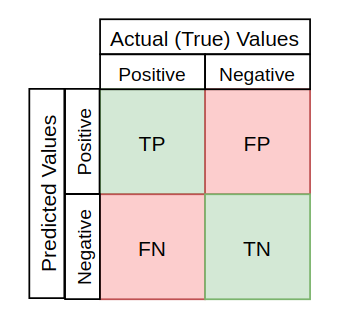


* **Recall** đề cập đến phần trăm tổng số kết quả có liên quan được phân loại chính xác theo thuật toán của bạn.

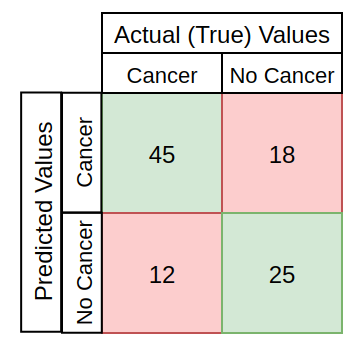


* 1. **Ma trận nhầm lẫn**

Một ma trận nhầm lẫn được sử dụng để minh họa hiệu suất của bộ phân loại dựa trên bốn giá trị trên (TP, FP, TN, FN). Chúng được vẽ với nhau để hiển thị một ma trận nhầm lẫn:



Để hiểu hơn ta bắt đầu với ví dụ dự đoán ung thư, một ma trận nhầm lẫn cho 100 bệnh nhân có thể trông giống như sau:



* Ví dụ này có:
* TP: Dự đoán đúng 45 trường hợp dương tính
* TN: Dự đoán đúng 25 trường hợp tiêu cực
* FP: 18 trường hợp tiêu cực bị phân loại sai (dự đoán dương tính sai)
* FN: 12 trường hợp dương tính bị phân loại sai (dự đoán tiêu cực sai)

1. **Phát biểu bài toán**

Xử lý ngôn ngữ tự nhiên chỉ là một phần của vấn đề. Những đoạn hội thoại giữa con người không chỉ đơn giản là những từ ngữ với ngữ nghĩa rõ ràng. Những đoạn văn còn mang nhiều sắc thái hoàn cảnh và rất phức tạp. Ví dụ, khi bạn của bạn đang buồn khổ, cáu giận hay tò mò, bạn sẽ có những cách hỏi chuyện khác nhau. Hay việc lựa chọn từ hoặc dấu câu theo các văn cảnh khác nhau sẽ có những ngữ nghĩa khác nhau. Hay việc bạn có thể dễ dàng hiểu được đánh giá thích hay không thích các bình luận review của một cuốn sách ngay cả khi họ không bao giờ nói trực tiếp ra.

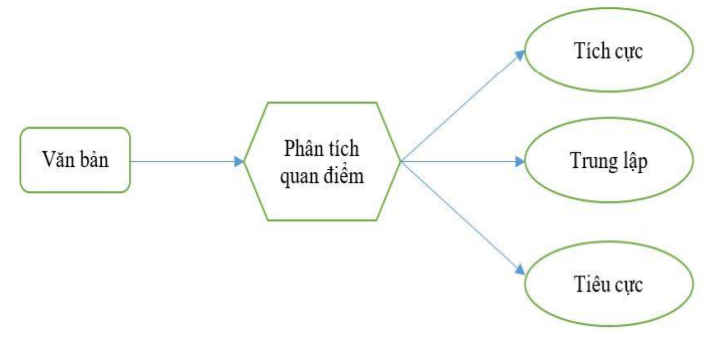
Nói chung, tình cảm là sự kết hợp tổng hợp của giọng điệu, sự lựa chọn từ ngữ và phong cách viết. Để máy tính thực sự hiểu được nội dung giao tiếp hàng ngày của con người, chúng cần được học nhiều hơn các định nghĩa khách quan của các từ, các tình cảm câu văn và ý nghĩa thực sự của chúng.

1. **Định nghĩa bài toán**

Các kỹ thuật xử lý ngôn ngữ tự nhiên rất hữu ích cho việc phân tích tình cảm. Nó thường được sử dụng để khám phá thông tin chi tiết về dữ liệu từ phương tiện truyền thông xã hội. Các công ty sử dụng phân tích tình cảm để hiểu về họ và các sản phẩm hoặc dịch vụ của họ.

NLP giúp xác định cảm xúc giữa một số bài đăng và nhận xét trực tuyến. Doanh nghiệp sử dụng các kỹ thuật NLP để biết ý kiến ​​của khách hàng về sản phẩm và dịch vụ của họ từ các đánh giá trực tuyến. Cụ thể là bài toán phân tích cảm xúc người dùng cho các sản phẩm, ứng dụng trên Google Play.

* **Đầu vào:** tập D = {d1, d2, d3, …, dN} gồm N [văn bản] bình luận của người dùng về các sản phẩm ứng dụng trên Google Play đã gắn nhãn tích cực, tiêu cực, và trung tính.
* **Đầu ra:** A = {a1, a2, a3} là tập quan điểm của người dùng.



**Hình:** Mô hình của bài toán phân tích quan điểm

1. **Cách tiếp cận**

Có hai cách tiếp cận để giải quyết bài toán phân tích tình cảm:

* Các thuật toán học máy (Machine learning algorithms)
* Xây dựng một mạng nơ-ron sâu (Build a deep neural network)

1. **Chuẩn bị dữ liệu**

Dữ liệu chứa hơn 12000 đánh giá về các ứng dụng cửa hàng ứng dụng khác nhau của người dùng thực. Dữ liệu cũng chứa xếp hạng do họ đưa ra để nó có thể được phân loại thành đánh giá tích cực, tiêu cực hoặc trung tính.

Các bài đánh thì:

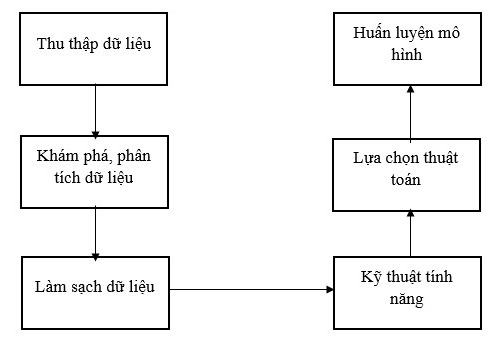
* Trực tiếp bày tỏ ý kiến của người dùng.
* Các bối cảnh khác nhau.
* Chứa lượng văn bản lớn và được cập nhật.

Xếp hạng các bài đánh giá:

* Đánh giá thuộc bốn và năm sao là cảm xúc tích cực
* Đánh giá thuộc ba sao là trung tính
* Đánh giá thuộc một và hai sao là cảm xúc tiêu cực

1. **Phân tích dữ liệu**

Để xây dựng được mô hình để giải quyết bài toán chúng ta cần phải áp dụng linh hoạt quy trình của việc xử lý ngôn ngữ tự nhên.



**Hình**. Quy trình tổng quan cho bài toán

1. **Thu thập dữ liệu**

Để bắt đầu với một dự án về xử lý ngôn ngữ tự nhiên việc đầu tiên cần làm đó là thu thập dữ liệu. Việc thu thập phải đáp ứng được các yêu cầu ở phần chuẩn bị dữ liệu. Dữ liệu thu thập càng lớn thì độ chính xác của mô hình càng cao.

1. **Phân tích dữ liệu khám phá (EDA)**

Sau khi thu thập dữ liệu, chúng ta tiến hành khám phá dữ liệu để hiểu về dữ liệu bằng cách tóm tắt các đặc điểm chính của chúng thường là vẽ biểu đồ trực quan. Thông qua quá trình EDA, có thể yêu cầu xác định câu lệnh hoặc định nghĩa vấn đề trên tập dữ liệu.

* Đối với tập dữ liệu về các bài đánh giá trên Google Play thì chúng ta cần:
* Kiểm tra các loại dữ liệu
* Bỏ các cột không liên quan
* Bỏ các hàng trùng lặp
* Kiểm tra các giá trị bị thiếu hoặc rỗng
* Khám phá dữ liệu mục tiêu
* Chuyển đổi các sao đánh giá thành phân loại tình cảm

1. **Tiền xử lý dữ liệu**

Làm sạch dữ liệu: trước khi bắt đầu với bất kỳ dự án NLP nào, chúng ta cần xử lý trước dữ liệu để có được tất cả ở định dạng nhất quán. Chúng ta cần làm sạch, mã hóa (token) và chuyển đổi dữ liệu của mình thành ma trận. Một số kỹ thuật xử lý trước văn bản cơ bản áp dụng cho việc giải quyết bài toán bao gồm:

* Xóa biểu tượng cảm xúc và xóa tất cả các ký tự đặc biệt và không cần thiết.
* Xóa các khoảng trắng thừa.
* Chuyển văn bản thành chữ thường
* Xóa từ dừng - stop words.
* Xóa freqwords
* Chuẩn hóa mã thông báo
* Chuyển đổi từ trò chuyện

1. **Kỹ thuật tính năng**

Ở bước này, dựa vào những phân tích và xử lý chúng ta lựa chọn các tính năng phù hợp sau đó chuyển đổi văn bản thành một vector có ý nghĩa.

* Sử dụng thuật toán Bag of Words với hai hướng tiếp cận:
* CountVectorizer
* TfidfVectorizer

1. **Lựa chọn thuật toán**

* Thuật toán máy học
* Decision Tree
* Random Forest
* Logistic Regression
* Thuật toán học sâu
* Bidirectional Encoder Representations - BERT

1. **Huấn luyện mô hình**

* Chia dữ liệu để huấn luyện mô hình. Trình bày kết quả thử nghiệm trên mô hình.

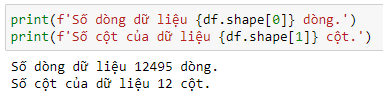
1. **Thực nghiệm và kết quả**
2. **Sử dụng các thuật toán học máy**
   1. **Cài đặt các thư viện cần thiết và đọc dữ liệu lên dataframe**



* Đọc dữ liệu từ bộ dữ liệu (dataset)



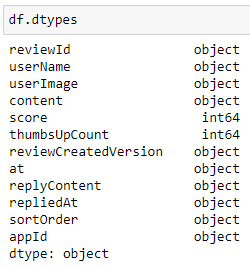




* 1. **Phân tích dữ liệu khám phá (EDA)**

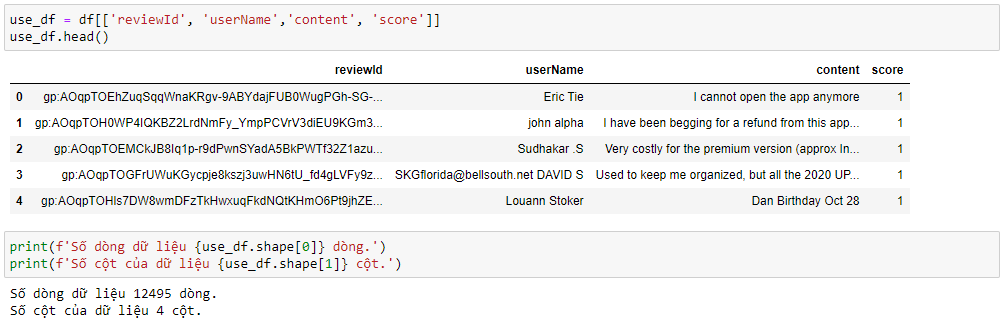
1. **Kiểm tra loại dữ liệu**

* Hầu hết dữ liệu có thể được phân loại thành 4 loại cơ bản theo quan điểm Học máy: dữ liệu số, dữ liệu phân loại, dữ liệu chuỗi thời gian, chữ.
* Hiểu các loại dữ liệu khác nhau có thể giúp chúng ta xác định các kỹ thuật tiền xử lý chính xác và chuyển đổi dữ liệu một cách thích hợp.



1. **Bỏ các cột không liên quan**

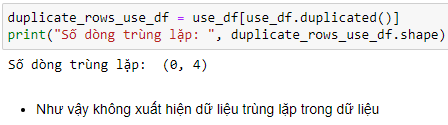
* Bước này chắc chắn là cần thiết trong mọi EDA vì đôi khi sẽ có nhiều cột mà chúng tôi không bao giờ sử dụng trong những trường hợp như vậy, giải pháp duy nhất là loại bỏ đi.



**Hình.** Giữ các cột dữ liệu có liên quan

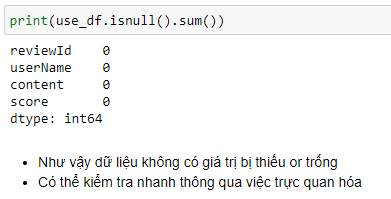
1. **Bỏ các hàng trùng lặp**

* Thường có một số dữ liệu trùng lặp có thể gây phiền hà không cung cấp thêm cho chúng ta thông tin mà còn ảnh hưởng xấu đến hiệu quả của mô hình đào tạo.



1. **Kiểm tra các giá trị bị thiếu hoặc rỗng**

* Kiểm tra các giá trị bị thiếu hoặc bị trống giúp loại bỏ những yếu tố không tốt cho việc train mô hình. Tùy vào trường hợp chúng ta có thể xóa, hoặc thay thế chúng bằng các giá trị (mean, median, mode)

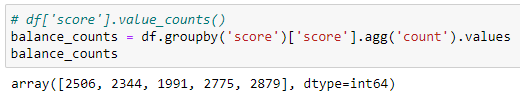


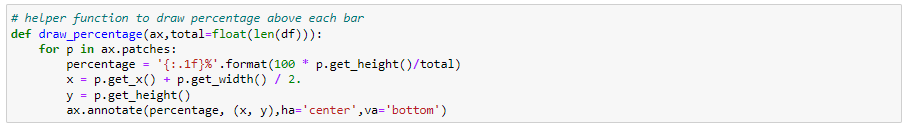
****

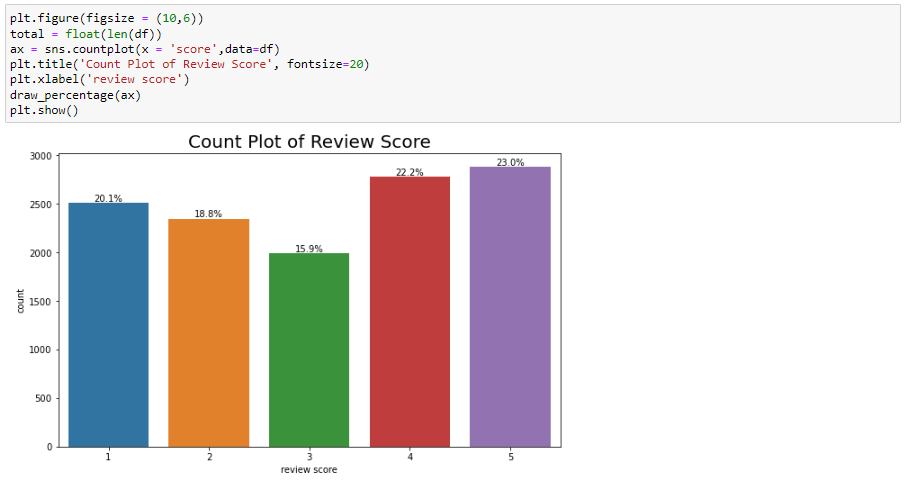
**Hình.** Code và biểu đồ kiểm tra giá trị thiếu hoặc rỗng

1. **Khám phá dữ liệu mục tiêu**

* Xem phân phối của các phân loại trong biến mục tiêu:

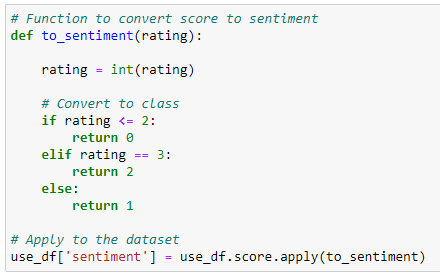


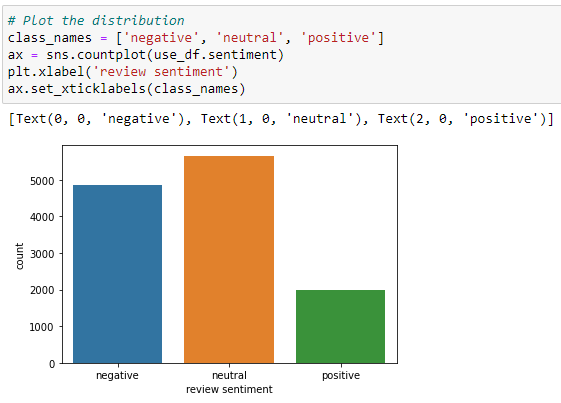




**Hình.** Code và biểu đồ biểu diễn các phân loại của biến mục tiêu

1. **Chuyển đổi điểm số (score) thành phân loại tình cảm**





**Hình.** Biểu đồ phân loại tình sau cảm sau chuyển đổi

* 1. **Tiền xử lý dữ liệu**

1. **Xóa biểu tượng cảm xúc và xóa tất cả các ký tự đặc biệt.**

Biểu tượng cảm xúc (**Emoticon**) là một dạng ngắn gọn của “Cảm xúc & Biểu tượng”. Nó là một biểu hiện của nét mặt như nụ cười hoặc cau mày, được hình thành bởi nhiều sự kết hợp khác nhau của các ký tự bàn phím và được sử dụng để truyền đạt cảm xúc hoặc giọng điệu dự định của người viết.

* Ví dụ:
* :-) Smile
* ;-) Hãy nháy mắt cười
* : <}) Người dùng có ria mép, mỉm cười
* : - || Mad
* :-( Sad
* : '- ( Crying: ~ Cũng khóc
* :-)) Hạnh phúc thật đấy: -DBig toe toét
* : - \* A kiss
* : -P ~ A liếm
* : -o Wow! hoặc tôi ngạc nhiên
* : - | Grim: - Lè lưỡi
* : - Người dùng tình cờ là Popeye
* : - / Bối rối
* =: O Sợ hãi (dựng tóc gáy)

Biểu tượng cảm xúc (**Emoji**) là một hình ảnh hoặc biểu tượng kỹ thuật số nhỏ được sử dụng để thể hiện một ý tưởng hoặc cảm xúc. Chúng đủ nhỏ để chèn vào văn bản. Trong tiếng Nhật “e” có nghĩa là hình ảnh và “moji” có nghĩa là ký tự.

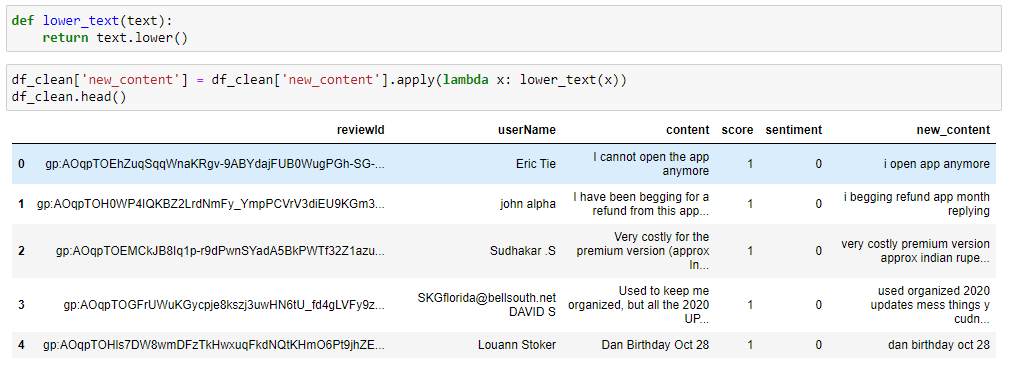


1. **Xóa các khoảng trắng thừa.**



1. **Chuyển văn bản thành chữ thường**

Chuyển văn bản về cùng một định dạng để dễ áp dụng các kỹ thuật xử lý ngôn ngữ tự nhiên.



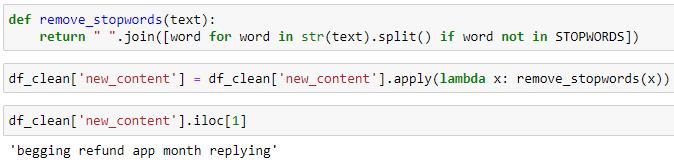
**Hình.** Code chuyển văn bản sang chữ thường

1. **Xóa từ dừng - stop words.**

Các từ như: a, an, the,… không cung cấp thông tin giá trị nhưng lại xuất hiện khá nhiều lần dẫn đến hiệu suất mô hình giảm.



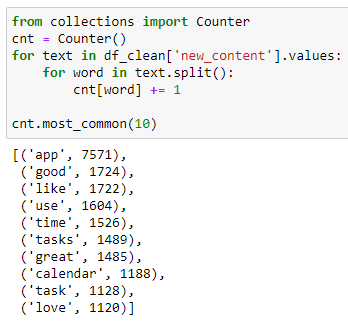
**Hình.** Danh sách các stop-words sử dụng thư viện spacy



**Hình.** Code thực hiện loại bỏ stop-words

1. **Xóa freqwords**

Nếu chúng ta có một kho ngữ liệu cụ thể về miền, chúng ta cũng có thể có một số từ thường gặp không quá quan trọng đối với chúng ta.



**Hình.** Code đếm các từ xuất hiện thường xuyên

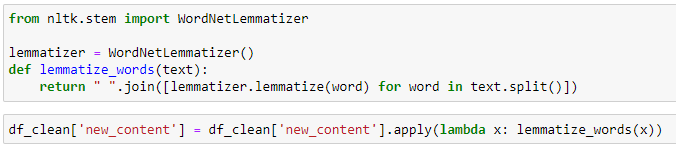


**Hình.** Code loại bỏ các từ xuất hiện thường xuyên

1. **Chuẩn hóa mã thông báo**

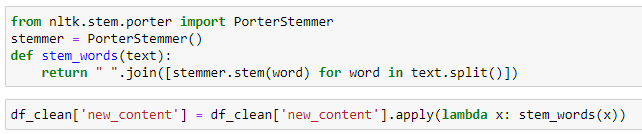
Chuẩn hóa mã thông báo có nghĩa là chuyển đổi các mã thông báo khác nhau sang dạng cơ sở của chúng. Điều này có thể được thực hiện bằng cách:

* **Lemmatization**: loại bỏ và thay thế các hậu tố để chuyển về dạng gốc của từ, được gọi là phần gốc.



**Hình.** Code xử lý Lemmatization

* **Stemming**: Trả về dạng cơ sở hoặc từ điển của một từ, được gọi là bổ đề.



**Hình.** Code xử lý Stemming

1. **Chuyển đổi từ trò chuyện**

Đây là bước tiền xử lý văn bản quan trọng nếu chúng ta đang xử lý dữ liệu trò chuyện. Mọi người sử dụng rất nhiều từ viết tắt trong trò chuyện và vì vậy có thể hữu ích nếu mở rộng những từ đó cho mục đích phân tích



**Hình.** Code xử lý chuyển đổi từ trò chuyện

* 1. **Vec-tơ hóa – Mô hình hóa – Đánh giá mô hình**

Sau giai đoạn tiền xử lý ban đầu, chúng ta cần biến đổi văn bản thành một vectơ (hoặc mảng) số có ý nghĩa.

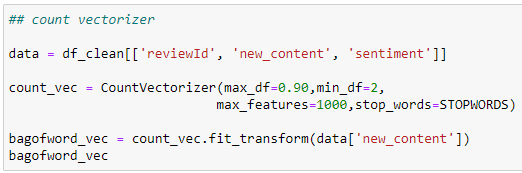
* **Bag of Words**

Túi từ là một bản trình bày văn bản mô tả sự xuất hiện của các từ trong tài liệu. Nó liên quan đến hai điều:

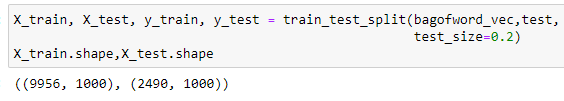
* Một vốn từ vựng của các từ đã biết.
* Một thước đo sự hiện diện của các từ đã biết.

Tại sao nó được gọi là “bag” các từ? Đó là bởi vì bất kỳ thông tin nào về thứ tự hoặc cấu trúc của các từ trong tài liệu đều bị loại bỏ và mô hình chỉ quan tâm đến việc liệu các từ đã biết có xuất hiện trong tài liệu hay không, chứ không phải nơi chúng xuất hiện trong tài liệu.

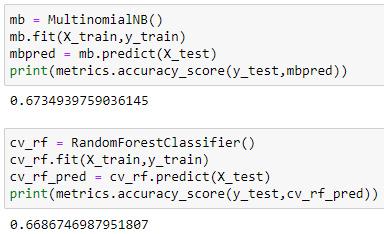
* **Countvectorizer:**



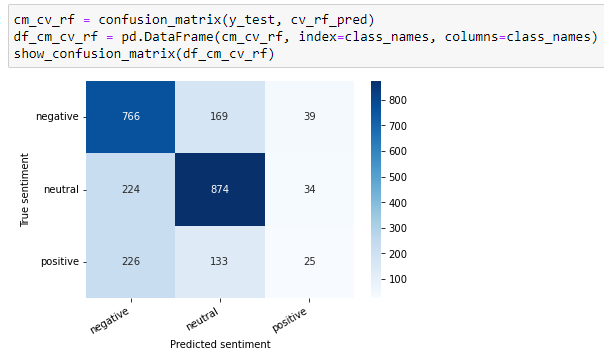
**Hình.** Code chuyển văn bản thành vector



**Hình.** Code chia tập dữ liệu thành dữ liệu huấn luyện và dữ liệu thử nghiệm



**Hình.** Code mô hình Multinomial Navie Bayes



**Hình.** Ma trận nhầm lẫn (confusion matrix) của mô hình Random Forest

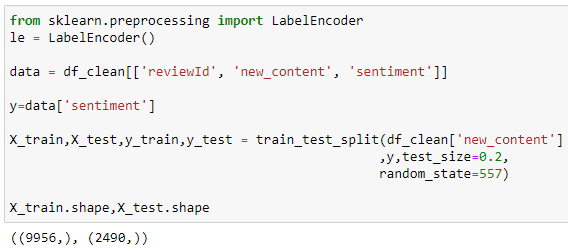
|  |  |  |
| --- | --- | --- |
|  | **MultinomialNB (%)** | **RandomForestClassifier (%)** |
| **Tiếng Anh** | 0.67 | 0.69 |
| **Tiếng Việt** | 0.67 | 0.70 |

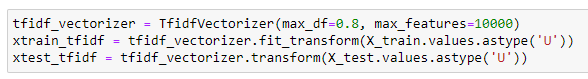
* Không có sự chênh lệch quá nhiều về độ chính xác đến từ 2 tập dữ liệu tiếng Anh và tiếng Việt.
* **TF-IDF**

Một vấn đề với phương pháp Bag of Words là các từ có tần suất cao bắt đầu chiếm ưu thế trong tài liệu (ví dụ: điểm lớn hơn), nhưng có thể không chứa nhiều “nội dung thông tin”. Ngoài ra, nó sẽ tăng trọng lượng cho các tài liệu dài hơn so với các tài liệu ngắn hơn.

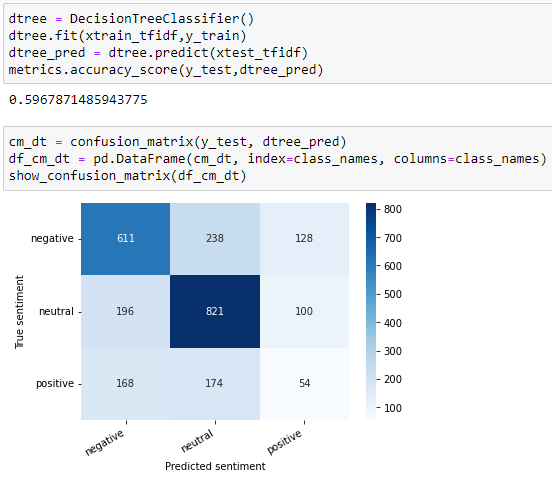
Một cách tiếp cận là bán lại tần suất của các từ theo tần suất chúng xuất hiện trong tất cả các tài liệu để điểm số cho các từ thường xuyên như “the” cũng thường xuyên xuất hiện trên tất cả các tài liệu sẽ bị phạt. Cách tiếp cận để tính điểm này được gọi là Tần suất tài liệu nghịch đảo thuật ngữ, viết tắt là TF-IDF, trong đó:

* Term Frequency: là điểm số của tần suất xuất hiện của từ trong tài liệu hiện tại.
* **TF = (Số lần thuật ngữ t xuất hiện trong tài liệu) / (Số thuật ngữ trong tài liệu)**
* Inverse Document Frequency:là điểm cho biết mức độ hiếm của từ này trong các tài liệu.
* **IDF = 1 + log (N / n), trong đó, N là số lượng tài liệu và n là số lượng tài liệu mà một thuật ngữ t đã xuất hiện.**

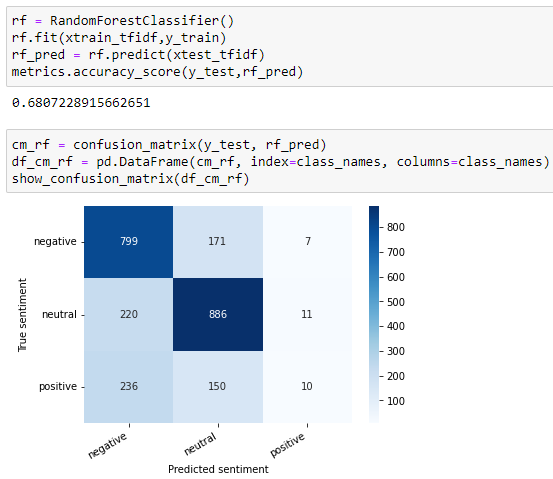




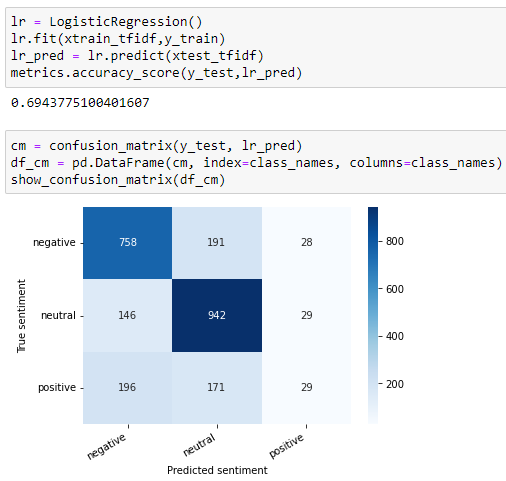
**Hình.** Code xử lý chuyển văn bản thành vector và chia tập dữ liệu huấn luyện và thử nghiệm



**Hình.** Ma trận nhầm lẫn mô hình Decision Tree Classification



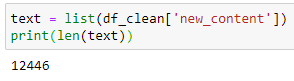
**Hình.** Ma trận nhầm lẫn mô hình Ramdom Forest Classification



**Hình.** Ma trận nhầm lẫn mô hình Logistic Regression

|  |  |  |  |
| --- | --- | --- | --- |
|  | **DecisionTreeClassifier (%)** | **RandomForestClassifier (%)** | **LogisticRegression (%)** |
| **Tiếng Anh** | 0.59 | 0.68 | 0.69 |
| **Tiếng Việt** | 0.65 | 0.72 | 0.72 |

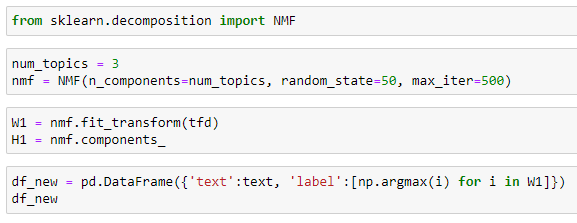
* Dữ liệu tiếng Việt cho ra độ chính xác cao so với dữ liệu tiếng Anh trong số 3 mô hình.
* **TF-DIF X NFM**
* Độ lớn của dữ liệu



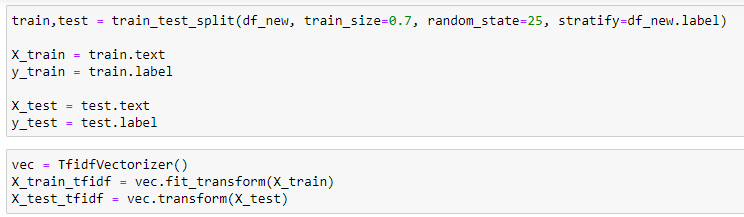
* Chuyển đổi văn bản thành vector sử dụng tf-idf



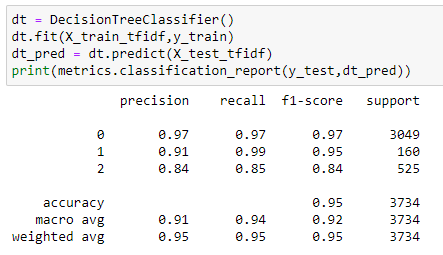
* Khởi tạo mô hình NMF - Non-negative Matrix-Factorization



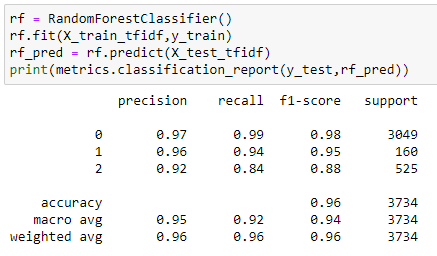
**Hình.** Khởi tạo mô hình NMF và gán nhãn



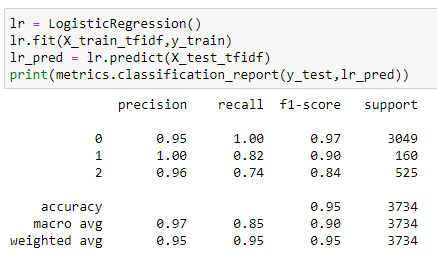
**Hình.** Code chia tập dữ liệu và chuyển đổi văn bản thành vector



**Hình.** Mô hình Decision Tree Classification và độ chính xác mô hình



**Hình 41.** Mô hình Random Forest Classification và độ chính xác



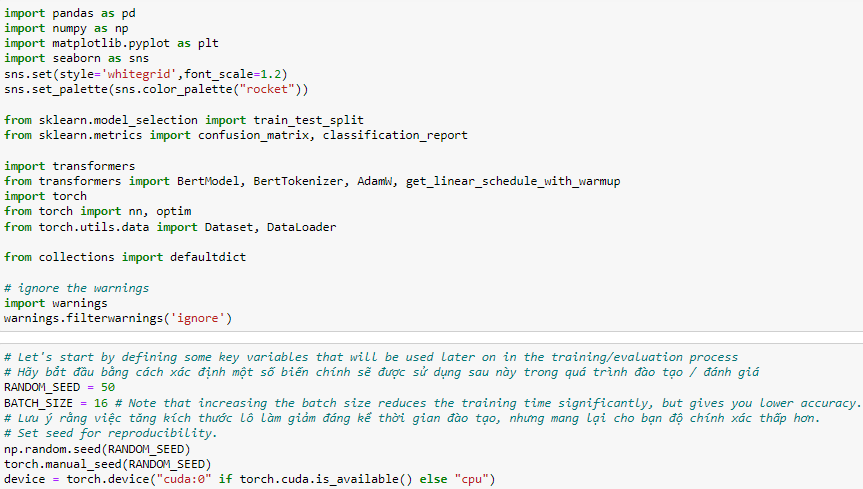
**Hình 42.** Mô hình Logistic Regression và độ chính xác



**Hình.** Ma trận nhầm lẫn mô hình Random Forest Classification

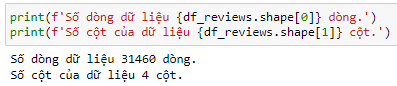
* Không có sự chênh lệch quá nhiều về độ chính xác đến từ 2 tập dữ liệu tiếng Anh và tiếng Việt ở phương pháp này.

1. **Sử dụng deep learning** 
   1. **Cài đặt các thư viện cần thiết và đọc dữ liệu lên dataframe**



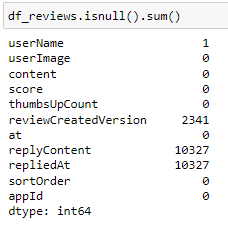
* Đọc dữ liệu từ bộ dữ liệu (dataset)

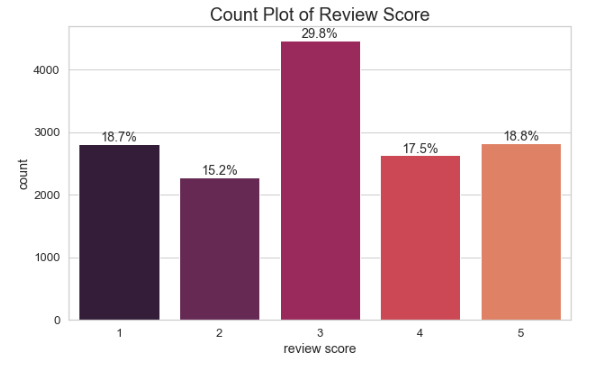




* 1. **Phân tích dữ liệu khám phá (EDA)**

1. **Kiểm tra các giá trị bị thiếu hoặc rỗng**



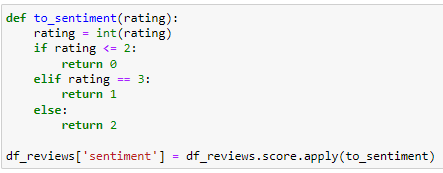


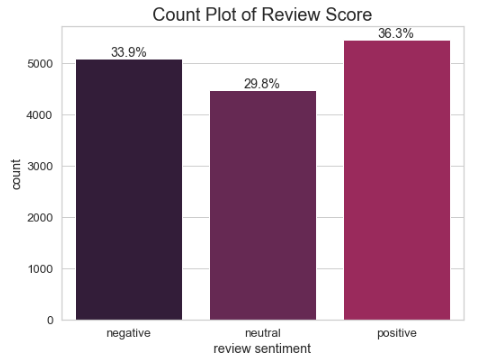
**Hình.** Biểu đồ xem xét sự cân bằng trong các lớp

Chúng ta có thể thấy rằng chúng ta có nhiều lớp tích cực hơn lớp tiêu cực và số lượng thấp của lớp trung lập. Giữ trung lập ít hơn để tập trung nhiều hơn vào các lớp tích cực và tiêu cực. Phân bổ các lớp dựa trên điểm số như sau:

* 0 - negative
* 1 - neutral
* 2 – positive

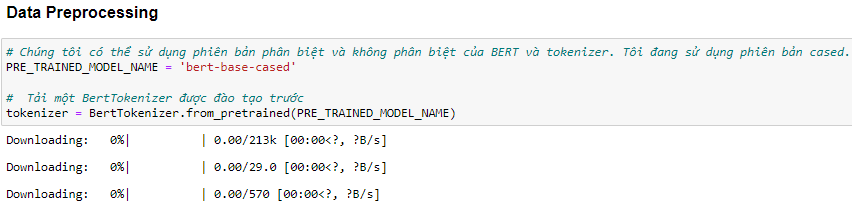
1. **Chuyển đổi điểm số (score) thành phân loại tình cảm**



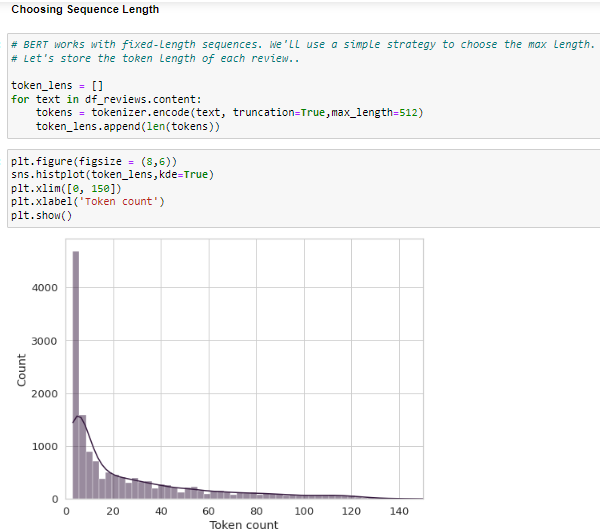


**Hình.** Biểu đồ xem xét sự cân bằng trong các lớp

* 1. **Tiền xử lý dữ liệu**
* Tải một mô hình đã được đào tạo trước để đào tạo cho dữ liệu của mình.

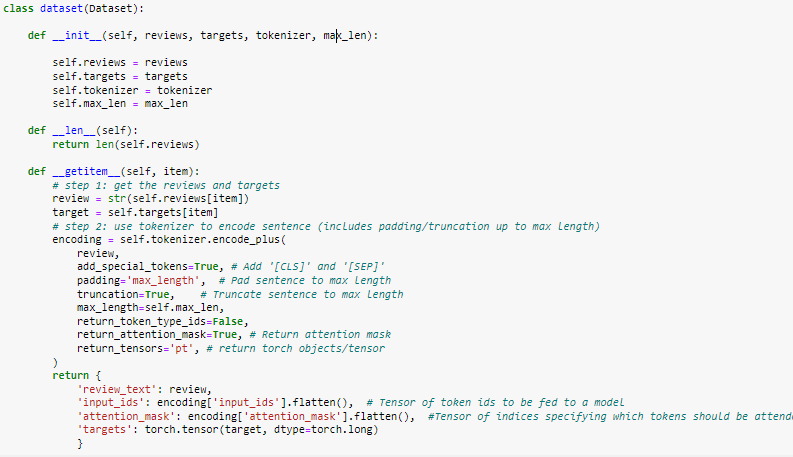


BERT hoạt động với các chuỗi có độ dài cố định. Chúng ta cần xác định độ dài tối đa. Lưu trữ độ dài mã thông báo của mỗi bài đánh giá.

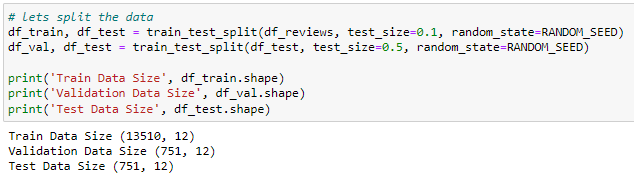




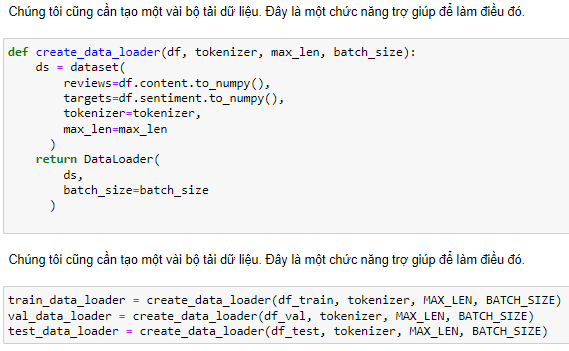
* 1. **Chuẩn bị dữ liệu**
* Tải bộ dữ liệu.



* 80% train data, 10% test và 10% validation data

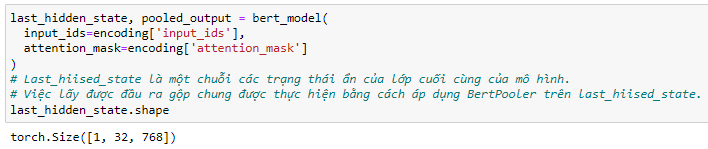


* Tạo một dataloader để giải phóng dữ liệu theo lô.

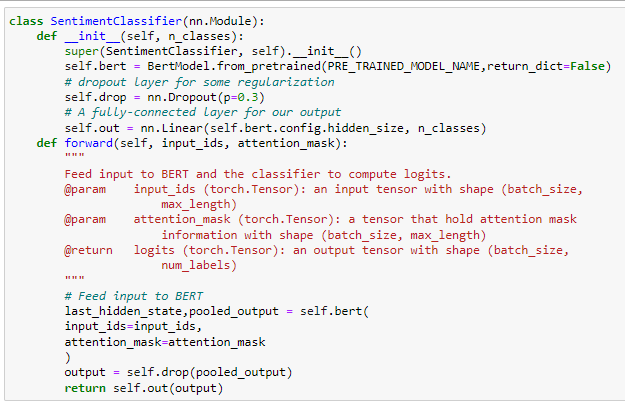


* 1. **Model Building – Xây dựng mô hình**
* Tải mô hình



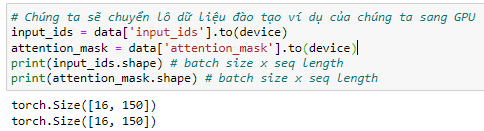


* Xây dựng lớp Phân loại tình cảm

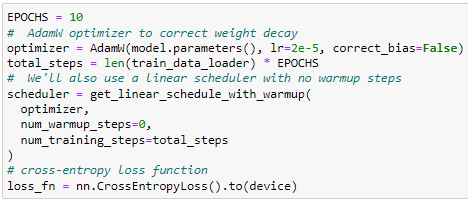


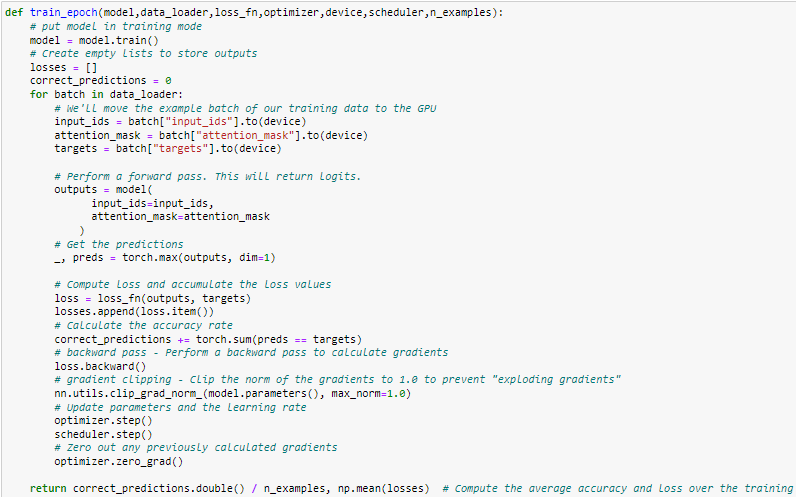
* Khởi tạo mô hình và chuyển sang trình phân loại



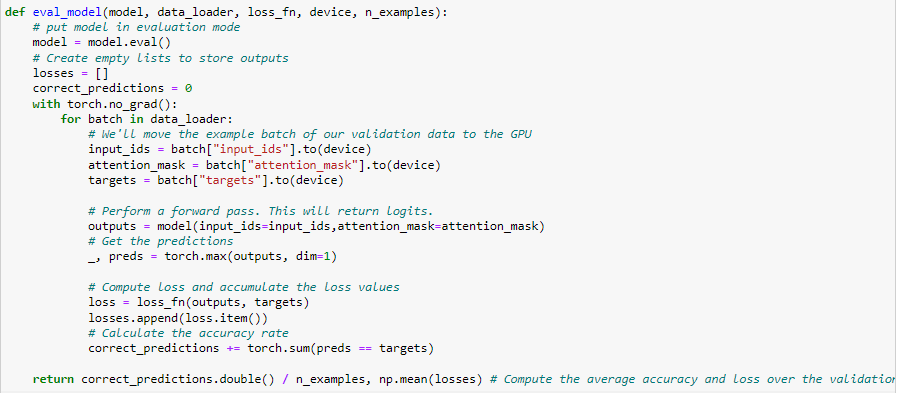


* Huấn luyện mô hình

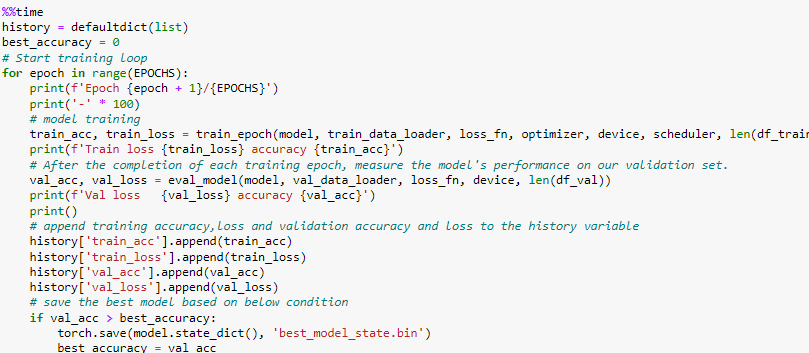




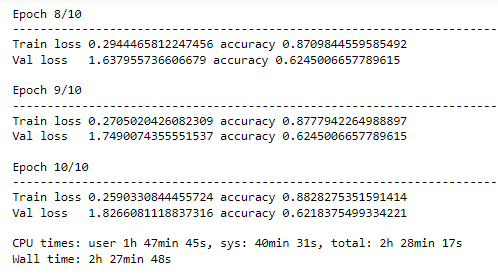
* Hàm để đánh giá hiệu suất của mô hình



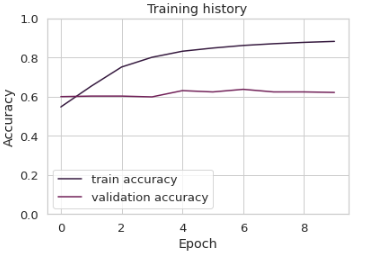
* Điểu chỉnh số lần đào tạo để tìm ra mô hình tối ưu nhất



* Kết quả sau 10 lần lặp



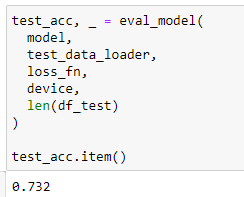
* Biểu đồ đào tạo và xác nhận độ chính xác.



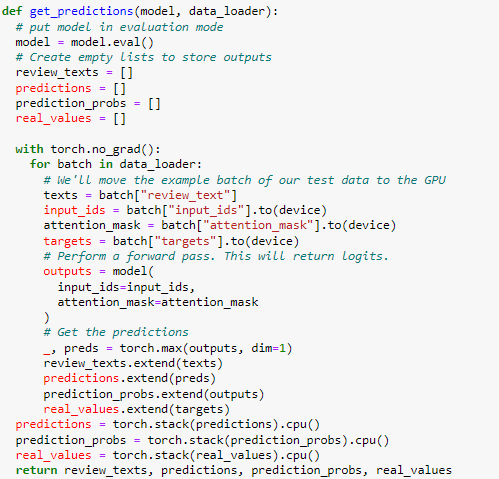
* Tải và sử dụng mô hình tối ưu nhất



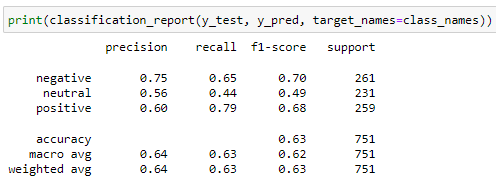
* 1. **Đánh giá mô hình**



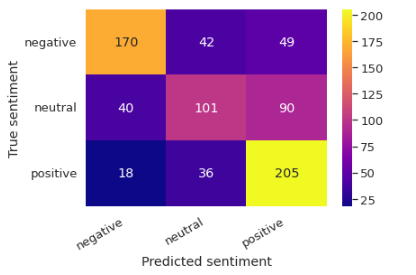
* Hàm để lấy dự đoán từ mô hình



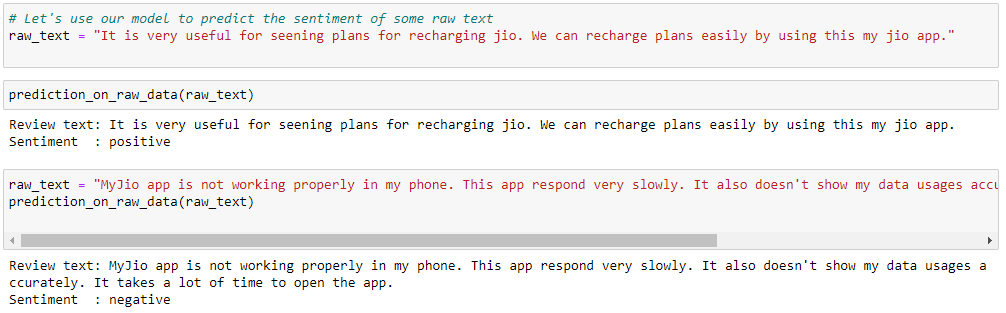
* Độ chính xác của dự đoán



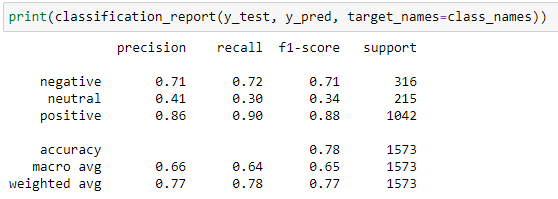
* Ma trận nhầm lẫn



* Kết quả dự đoán thử nghiệm

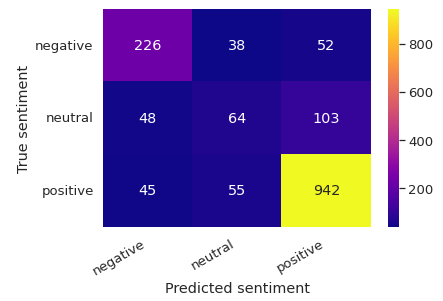


* 1. **So sánh với mô hình sử dụng bộ dữ liệu tiếng Việt**
* Độ chính xác của dự đoán

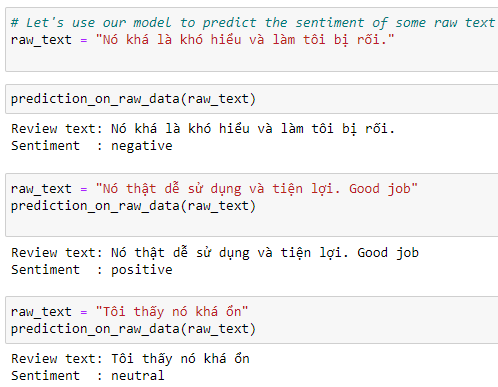


Độ chính xác 78% so với 63% trên bộ dữ liệu tiếng Anh.

* Ma trận nhầm lẫn

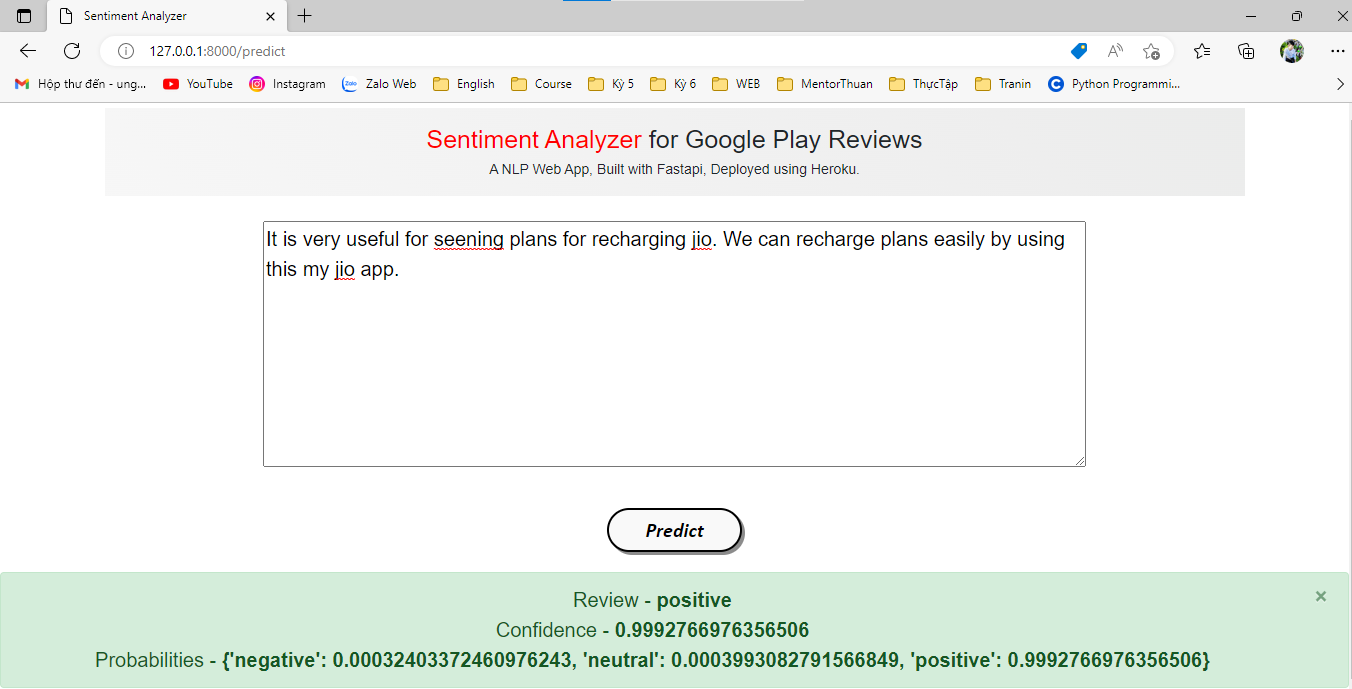


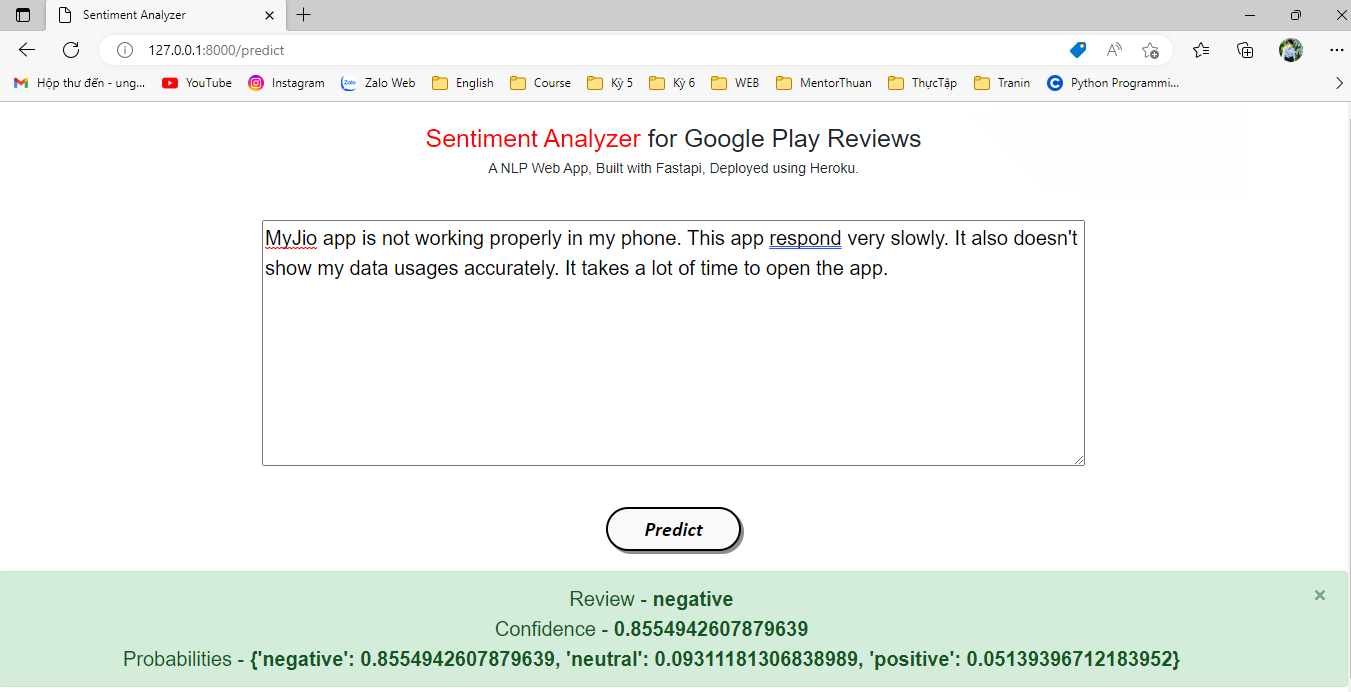
* Kết quả dự đoán thử nghiệm

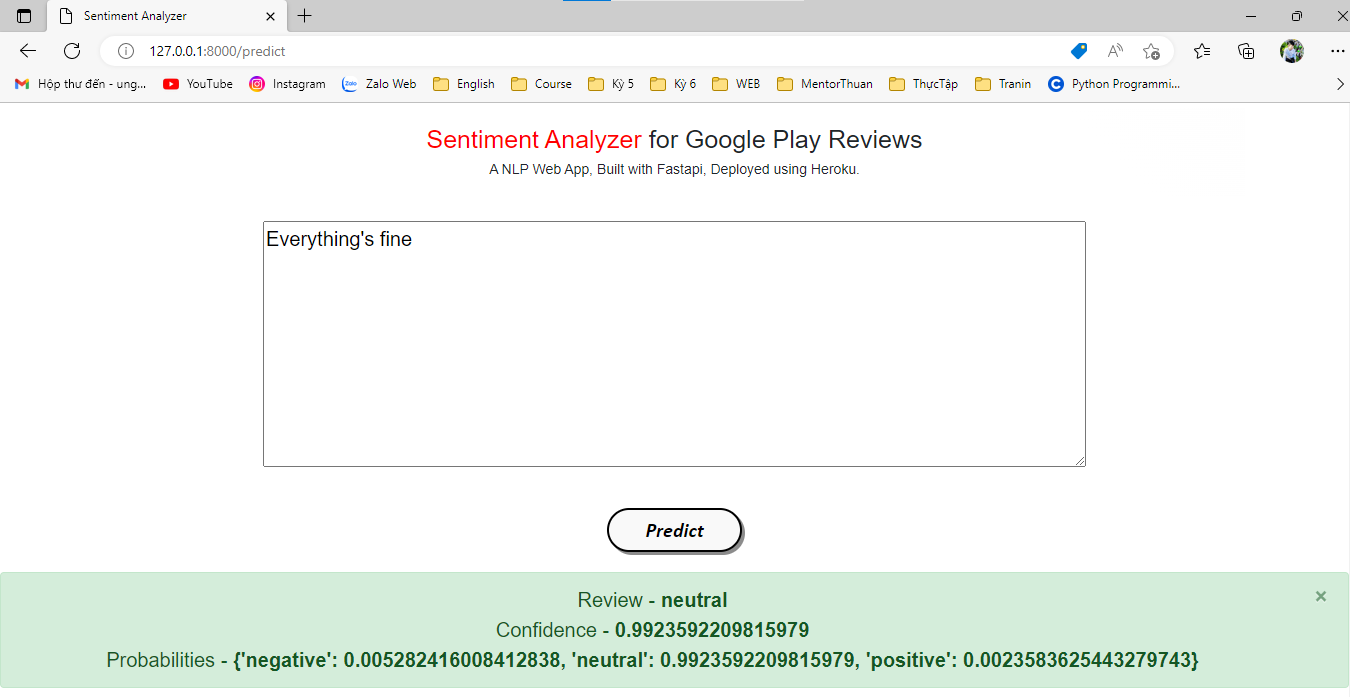


1. **Kết quả**

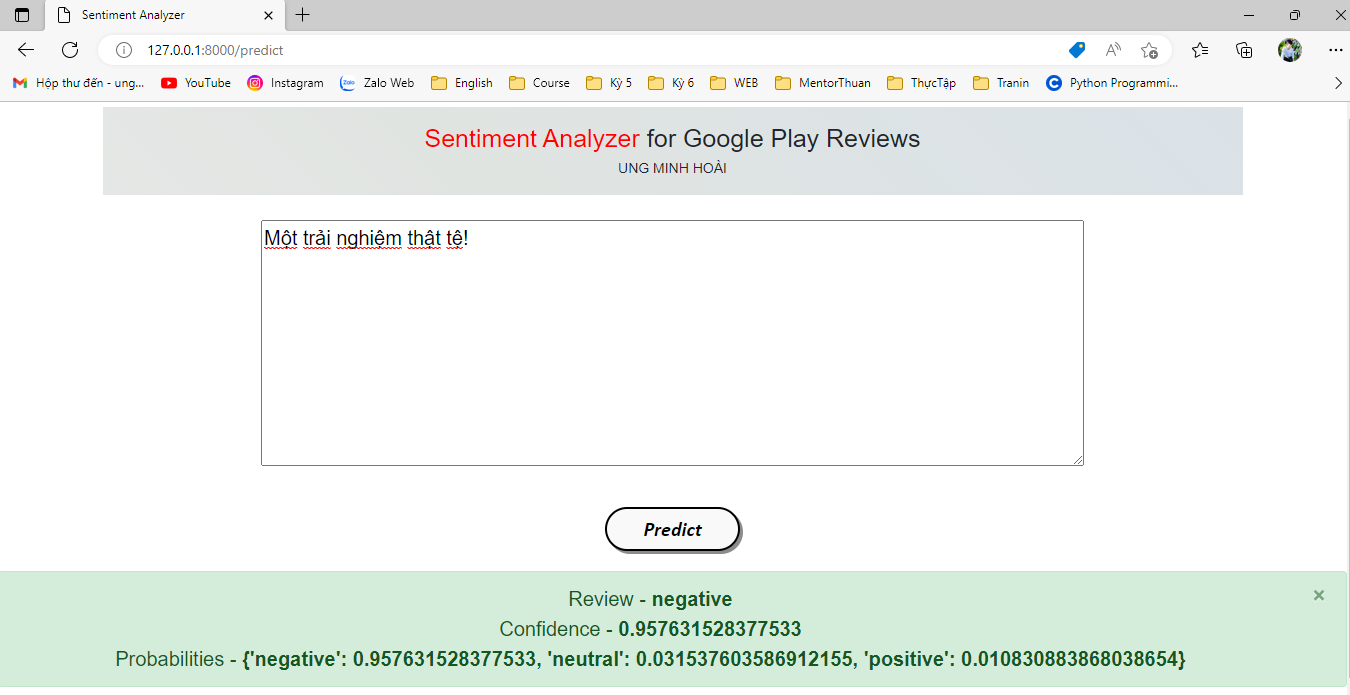
* Giao diện web thử nghiệm trên bộ dữ liệu tiếng Anh.



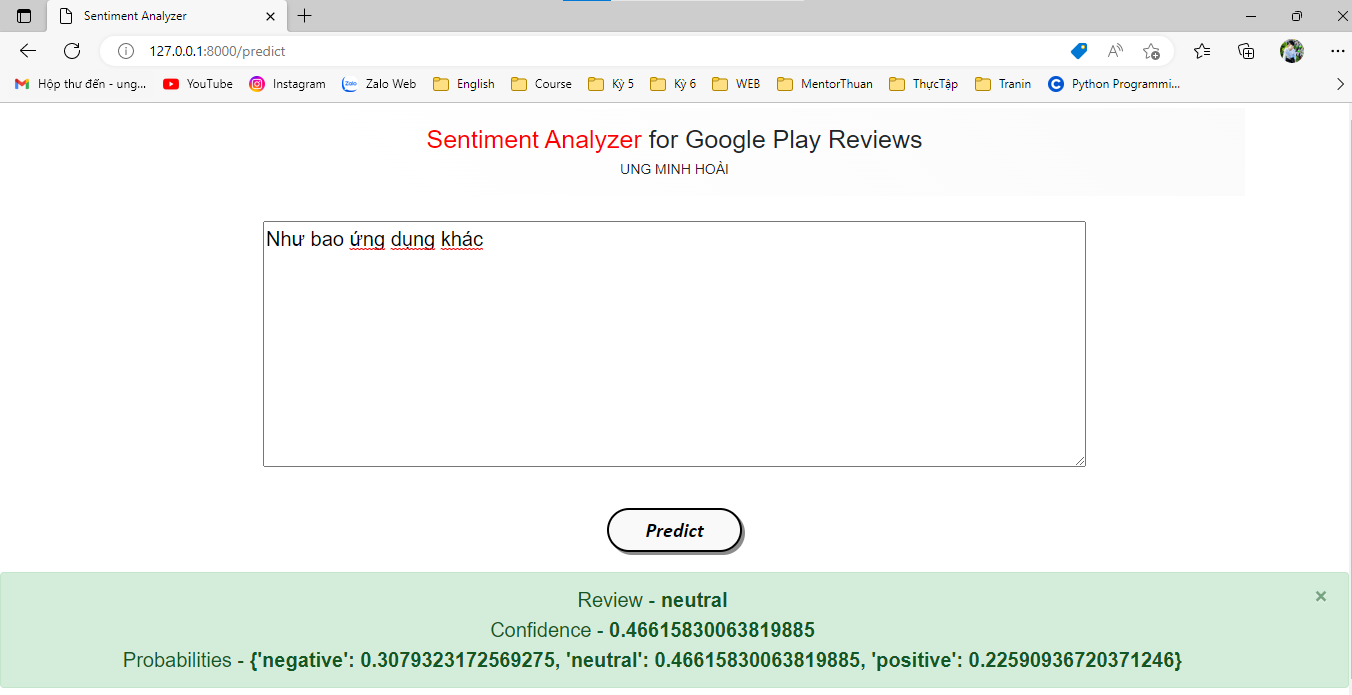




* Giao diện web thử nghiệm trên bộ dữ liệu tiếng Việt







1. **Kết luận**
2. **Những điểm đạt được**

* Hiểu và nắm được quy trình xử lý ngôn ngữ tự nhiên.
* Hiểu được bài toán của các chủ đề thường gặp trong xử lý ngôn ngữ tự nhiên như: mô hình ngôn ngữ, phân tích quan điểm, dịch máy, chuyển văn bản thành giọng nói,…
* Hiểu và vận dụng các thuật toán học máy như: Bag-of-words, Decision Tree, Random Forest, Logistic Regression.
* Hiểu và vận dụng các thuật toán học sâu như: RNN, LSTM, GRU, BERT, Transformer.
* Hiểu và vận dụng các chỉ số đánh giá độ chính xác mô hình vào bài toán.
* Thử nghiệm trên cả hai tập dữ liệu tiếng Anh và tiếng Việt.
* Tự code xử lý bài toán theo phương pháp học máy. Cải thiện độ chính xác của phương pháp học sâu.
* Trình bày khá chi tiết về các thuật toán sử dụng để giải quyết bài toán.
* Demo mô hình trên web app.

1. **Những điểm hạn chế**

* Độ chính xác chưa cao, cần cải thiện các làm số của mô hình.
* Chưa tự deploy được bài làm của mình lên web app mà sử dụng khuôn mẫu có sẵn.

1. **Hướng phát triển**

* Cải thiện hiệu suất, độ chính xác của mô hình.
* Tự deloy lên web app.

1. **Tài liệu tham khảo**
2. <https://github.com/ni3choudhary/Sentiment-Analysis-Google-Play-App-Reviews>
3. <http://jalammar.github.io/illustrated-transformer/>
4. <https://huggingface.co/models>
5. <https://www.kaggle.com/code/abhinand05/bert-for-humans-tutorial-baseline>
6. <https://blog.rosetta.ai/learn-hugging-face-transformers-bert-with-pytorch-in-5-minutes-acee1e3be63d>
7. <https://www.kaggle.com/code/prakharrathi25/sentiment-analysis-using-bert>
8. <https://towardsdatascience.com/nmf-a-visual-explainer-and-python-implementation-7ecdd73491f8>
9. <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>