

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CẦN THƠ
KHOA CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG**

**NIÊN LUẬN
NGÀNH CÔNG NGHỆ THÔNG TIN**

Đề tài

BÀI TOÁN PHÂN LOẠI CẢM XÚC

**Sinh viên: Lê Thái Nhật Nam
Mã số: B1809263
Khóa: K44**

Cần Thơ, 11/2021

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CẦN THƠ
KHOA CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG
BỘ MÔN CÔNG NGHỆ THÔNG TIN**

**NIÊN LUẬN CƠ SỞ
NGÀNH CÔNG NGHỆ THÔNG TIN**

Đề tài

BÀI TOÁN PHÂN LOẠI CẢM XÚC

**Người hướng dẫn
TS/Ths Lâm Nhật Khang**

**Sinh viên thực hiện
Lê Thái Nhật Nam
Mã số: B1809263
Khóa: K44**

Cần Thơ, 11/2021

MỞ ĐẦU

Trong những năm gần đây cùng với sự phát triển mạnh mẽ của nền kinh tế thị trường và sự phát triển vượt bậc của ngành Công nghệ thông tin thì nhiều doanh nghiệp hiện nay đã mở rộng sang một hình thức kinh doanh mới – “kinh doanh trực tuyến”. Hình thức kinh doanh này rất thuận tiện cho khách hàng cũng như thuận tiện cho việc tiếp cận số lượng lớn khách hàng. Doanh nghiệp có thể dễ dàng nắm bắt độ hài lòng của khách hàng qua các bình luận phản hồi về sản phẩm để cải thiện dịch vụ ngày càng hoàn thiện. Sẽ không có vấn đề gì nếu số lượng phản hồi từ khách hàng thấp. Tuy nhiên, việc kinh doanh ngày càng phát triển, lượng phản hồi quá lớn thì vấn đề nắm bắt cảm xúc của người dùng đối với dịch vụ của doanh nghiệp lại trở thành một vấn đề nan giải. Từ đó, ”Bài toán phân loại cảm xúc” qua các bình luận được đặt ra.

Vì lý do đó, niên này trình bày các giải thuật dựa trên lý thuyết máy học và mạng neural để đưa ra các mô hình nhằm phân tích cảm xúc cho các dữ liệu bình luận. Mục tiêu nghiên cứu niên luận sẽ đi sâu tìm cấu trúc của mạng neural trong việc xử lý, phân tích cảm xúc khách hàng dựa trên các bình luận phản hồi của họ, đồng thời dựa vào phân tích để phân loại các chia sẻ thành các loại (0 - negative – tệ, 1 – positive – tốt). Nhằm cho ra kết quả đánh giá có độ chính xác gần bằng với kết quả ban đầu đã được đánh giá bởi con người. Phạm vi nghiên cứu tập trung vào xây dựng các mô hình mạng neural trên Thư viện Keras bằng ngôn ngữ Python nhằm hiện thực hoá mô hình được nghiên cứu. Cụ thể hơn là mô hình *LSTM (Long Short Term Memory)* – Mô hình phát triển từ mô hình RNN và mô hình *Transformer*. Từ đó đánh giá độ chính xác của từng mô hình mạng neural được nghiên cứu dựa trên các dữ liệu thử nghiệm.

MỤC LỤC

MỞ ĐẦU	3
PHẦN I: MÔ HÌNH LSTM - LONG SHORT TERM MEMORY	7
CHƯƠNG I: TÌM HIỂU MÔ HÌNH LSTM	7
I. Mô hình LSMT là gì?	7
II. Bên trong một tế bào LSTM	8
CHƯƠNG II: XÂY DỰNG MÔ HÌNH LSTM TRONG PHÂN LOẠI CẢM XÚC	10
I. Xử lý dữ liệu	11
1. Chuẩn hóa dữ liệu.....	11
1.1 Chuẩn hóa dữ liệu với bộ dữ liệu bình luận bằng Tiếng Anh	11
1.1.1. Thông tin bộ dữ liệu bình luận bằng tiếng Anh.....	11
1.2.2. Chuẩn hóa dữ liệu Tiếng Anh:.....	11
1.2. Chuẩn hóa dữ liệu với bộ dữ liệu bình luận bằng Tiếng Việt	11
1.2.1. Thông tin bộ dữ liệu bình luận bằng tiếng Việt.....	11
1.2.2. Chuẩn hóa dữ liệu Tiếng Việt.....	12
2. Xử lý tách từ.....	13
3. Xây dựng bộ từ vựng từ tập các câu train và test.....	13
4. Số hóa các từ trong câu	13
II. Đưa các câu đã số hóa vào Embedding.....	13
1.Embedding là gì?.....	13
2. Ý nghĩa của việc đưa dữ liệu qua Embedding.....	13
III. Mô hình LSTM – Long Short Term Memory networks	13
IV. Lớp dự đoán thông thường Dense	13
1. Dense layer là gì?	13
2. Tác dụng của lớp Dense trong mô hình	14
CHƯƠNG III: THỰC NGHIỆM MÔ HÌNH LSTM TRONG BÀI TOÁN PHÂN LOẠI CẢM XÚC	15
I. Với bộ dữ liệu Tiếng Anh.....	15
1. Huấn luyện mô hình	15
2. Đánh giá mô hình với bộ dữ liệu Tiếng Anh	18
2.1. Sử dụng mô hình đánh giá Confusion matrix.....	18
2.2. Đánh giá với một câu mới	19
II. Với bộ dữ liệu Tiếng Việt	19
1. Huấn luyện mô hình	19
2. Đánh giá mô hình	20

2.1. Sử dụng mô hình đánh giá Confusion Matrix	20
2.2. Đánh giá mô hình với câu mới	20
PHẦN II: MÔ HÌNH TRANSFORMERS	21
CHƯƠNG I: TÌM HIỂU MÔ HÌNH TRANSFORMER TRONG BÀI TOÁN	
PHÂN LOẠI CẢM XÚC	21
I. Mô hình Transformer là gì?	21
II. Transformer trong bài toán phân loại cảm xúc	21
1. Positional Encoding	22
2. Self-Attention	22
3. Multi-head Attention	24
TÀI LIỆU THAM KHẢO	25

MỤC LỤC HÌNH

Hình 1: Mô hình mạng nơ-ron của LSTM	7
Hình 2: Tầng 1 trong mỗi tế bào của mô hình LSTM.....	8
Hình 3: Tầng 2 trong mỗi tế bào của mô hình LSTM.....	8
Hình 4: Tầng 3 trong mỗi tế bào của mô hình LSTM.....	9
Hình 5: Tầng 4 trong mỗi tế bào của mô hình LSTM.....	9
Hình 6: Sơ đồ mô hình LSTM trong phân loại cảm xúc	10
Hình 7: Sơ đồ xử lý dữ liệu đầu vào.....	10
Hình 8: Confusion Matrix cho mô hình LSTM với Tiếng Anh	18
Hình 9: Cấu trúc của mô hình Transformer	21
Hình 10: Positional Encoding.....	22
Hình 11: 3 vector Q,K,V của mỗi từ	22
Hình 12: Các bước tính Attention của một từ	23

PHẦN I: MÔ HÌNH LSTM - LONG SHORT TERM MEMORY

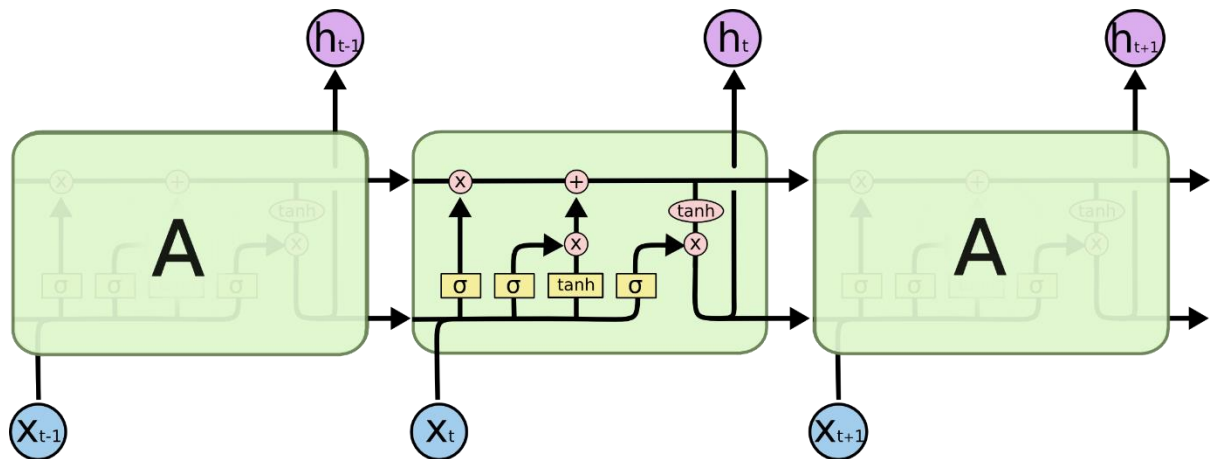
CHƯƠNG I: TÌM HIỂU MÔ HÌNH LSTM

I. Mô hình LSMT là gì?

Mạng bộ nhớ dài-ngắn (Long Short Term Memory networks), thường được gọi là LSTM - là một dạng đặc biệt của RNN, nó có khả năng học được các phụ thuộc xa. LSTM được giới thiệu bởi *Hochreiter & Schmidhuber (1997)*, và sau đó đã được cải tiến và phổ biến bởi rất nhiều người trong ngành. Chúng hoạt động cực kì hiệu quả trên nhiều bài toán khác nhau nên dần đã trở nên phổ biến như hiện nay.

LSTM được thiết kế để giải quyết vấn đề phụ thuộc xa (long-term dependency). Việc nhớ thông tin trong suốt thời gian dài là đặc tính mặc định của chúng, chứ ta không cần phải huấn luyện nó để có thể nhớ được. Tức là ngay nội tại của nó đã có thể ghi nhớ được mà không cần bất kì can thiệp nào.

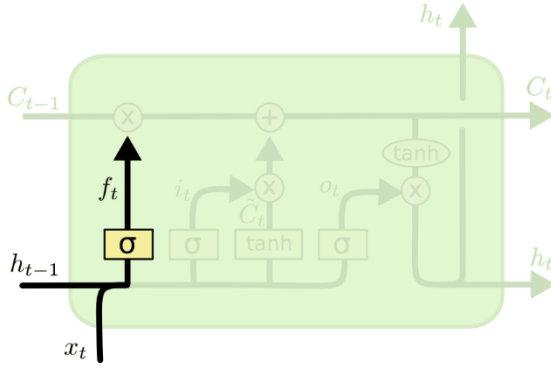
Mọi mạng hồi quy đều có dạng là một chuỗi các mô-đun lặp đi lặp lại của mạng nơ-ron. LSTM cũng có kiến trúc dạng chuỗi như vậy, nhưng các mô-đun trong nó có cấu trúc khác với mạng RNN chuẩn. Thay vì chỉ có một tầng mạng nơ-ron, chúng có tới 4 tầng tương tác với nhau một cách rất đặc biệt.



Hình 1: Mô hình mạng nơ-ron của LSTM
(Nguồn: <https://colah.github.io/>)

II. Bên trong một tế bào LSTM

Tầng đầu tiên của LSTM là quyết định xem thông tin nào cần bỏ đi từ trạng thái tế bào. Quyết định này được đưa ra bởi tầng sigmoid - gọi là “tầng cổng quên” (forget gate layer). Nó sẽ lấy đầu vào là h_{t-1} và x_t rồi đưa ra kết quả là một số trong khoảng $[0,1]$ cho mỗi số trong trạng thái tế bào C_{t-1} . Đầu ra là 1 thể hiện rằng thông tin được giữ lại, còn 0 tức là thông tin sẽ bị bỏ đi.

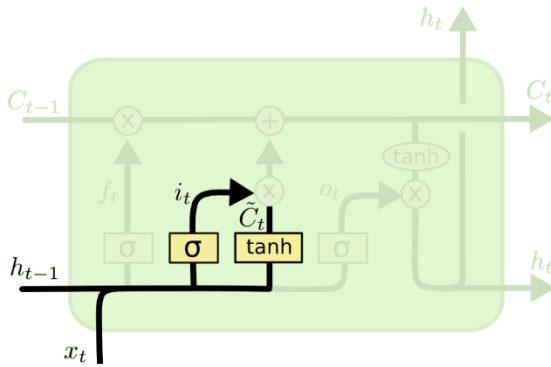


$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

Hình 2: Tầng 1 trong mỗi tế bào của mô hình LSTM

(Nguồn: <https://colah.github.io/>)

Tầng tiếp theo là quyết định xem thông tin mới nào ta sẽ lưu vào trạng thái tế bào. Việc này gồm 2 phần. Đầu tiên là sử dụng một tầng *sigmoid* được gọi là “tầng cổng vào” (input gate layer) để quyết định giá trị nào ta sẽ cập nhập. Tiếp theo là một tầng *tanh* tạo ra một véc-tơ cho giá trị mới \tilde{C}_t nhằm thêm vào cho trạng thái. Trong bước tiếp theo, ta sẽ kết hợp 2 giá trị đó lại để tạo ra một cập nhập cho trạng thái.



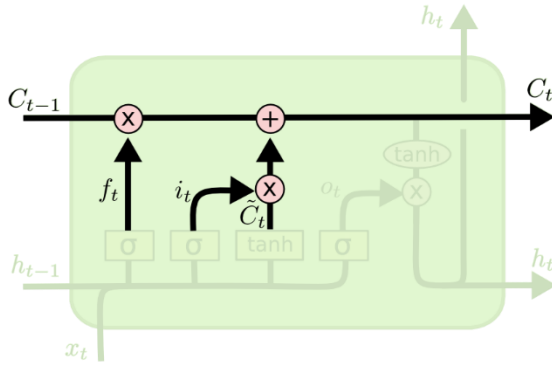
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Hình 3: Tầng 2 trong mỗi tế bào của mô hình LSTM

(Nguồn: <https://colah.github.io/>)

Tầng này là lúc cập nhập trạng thái tế bào cũ C_{t-1} thành trạng thái mới C_t . Ta sẽ nhân trạng thái cũ với f_t để bỏ đi những thông tin ta quyết định quên lúc trước. Sau đó cộng thêm $i_t * \tilde{C}_t$.

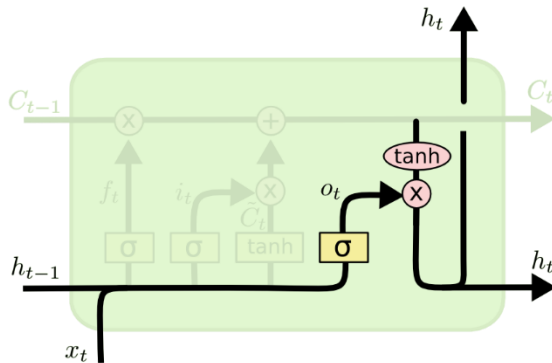


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Hình 4: Tầng 3 trong mỗi tế bào của mô hình LSTM

(Nguồn: <https://colah.github.io/>)

Cuối cùng, ta cần quyết định xem ta muốn đầu ra là gì. Giá trị đầu ra sẽ dựa vào trạng thái tế bào, nhưng sẽ được tiếp tục sàng lọc. Đầu tiên, ta chạy một tầng *sigmoid* để quyết định phần nào của trạng thái tế bào ta muốn xuất ra. Sau đó, ta đưa trạng thái tế bào qua một hàm *tanh* để cho giá trị nó về khoảng $[-1,1]$, và nhân nó với đầu ra của công *sigmoid* để được giá trị đầu ra ta mong muốn.



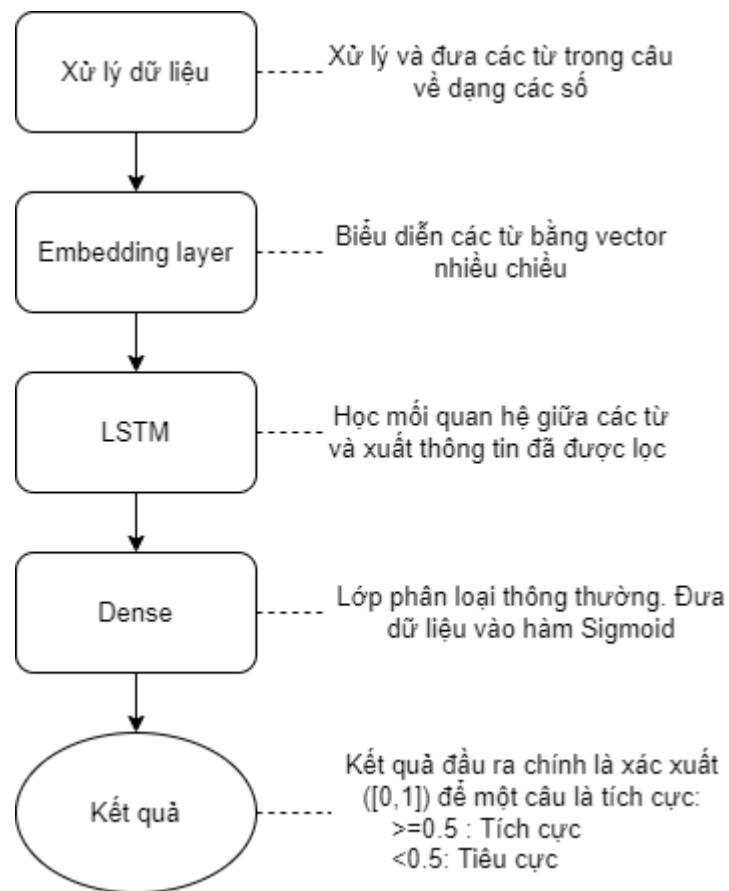
$$o_t = \sigma(W_o [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

Hình 5: Tầng 4 trong mỗi tế bào của mô hình LSTM

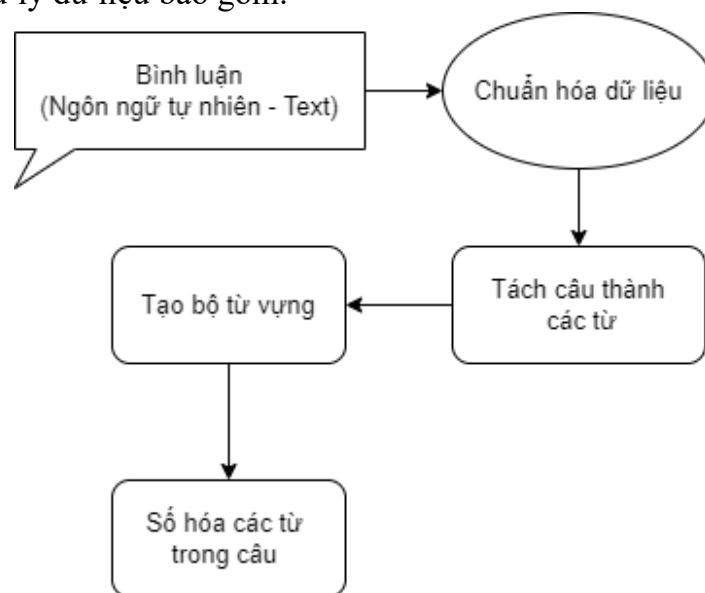
(Nguồn: <https://colah.github.io/>)

CHƯƠNG II: XÂY DỰNG MÔ HÌNH LSTM TRONG PHÂN LOẠI CẢM XÚC



Hình 6: Sơ đồ mô hình LSTM trong phân loại cảm xúc

Trong đó Xử lý dữ liệu bao gồm:



Hình 7: Sơ đồ xử lý dữ liệu đầu vào

I. Xử lý dữ liệu

1. Chuẩn hóa dữ liệu

1.1 Chuẩn hóa dữ liệu với bộ dữ liệu bình luận bằng Tiếng Anh

1.1.1. Thông tin bộ dữ liệu bình luận bằng tiếng Anh

- Dữ liệu sử dụng là bình luận về phim có sẵn trên thư viện của Tensorflow với tên bộ dữ liệu là: “imdb_reviews”

- Thông tin và cấu trúc của bộ dữ liệu:

```
tfds.core.DatasetInfo(  
    name='imdb_reviews',  
    version=1.0.0,  
    description='Large Movie Review Dataset.  
This is a dataset for binary sentiment classification containing substantially  
homepage=http://ai.stanford.edu/~amaas/data/sentiment/,  
    features=FeaturesDict({  
        'label': ClassLabel(shape=(), dtype=tf.int64, num_classes=2),  
        'text': Text(shape=(), dtype=tf.string),  
    }),  
    total_num_examples=100000,  
    splits={  
        'test': 25000,  
        'train': 25000,  
        'unsupervised': 50000,  
    },  
    supervised_keys=('text', 'label'),  
    citation="""@InProceedings{maas-EtAl:2011:ACL-HLT2011,  
    author   = {Maas, Andrew L. and Daly, Raymond E. and Pham, Peter T.  
    title    = {Learning Word Vectors for Sentiment Analysis},  
    booktitle = {Proceedings of the 49th Annual Meeting of the Association f  
    month    = {June},  
    year     = {2011},  
    address  = {Portland, Oregon, USA},  
    publisher = {Association for Computational Linguistics},  
    pages    = {142--150},  
    url      = {http://www.aclweb.org/anthology/P11-1015}  
    }""",  
    redistribution_info=,  
)
```

1.2.2. Chuẩn hóa dữ liệu Tiếng Anh:

- Xóa các ký tự đặc biệt.
- Xóa dấu cách đầu và cuối mỗi câu.
- Đảm bảo giữa mỗi từ chỉ bao gồm một dấu “khoảng trắng” bằng cách thay các “khoảng trắng” đứng cạnh nhau thành một khoảng trắng duy nhất.

(Ghi chú: Bộ dữ liệu trên đã được chuẩn hóa bởi tác giả).

1.2. Chuẩn hóa dữ liệu với bộ dữ liệu bình luận bằng Tiếng Việt

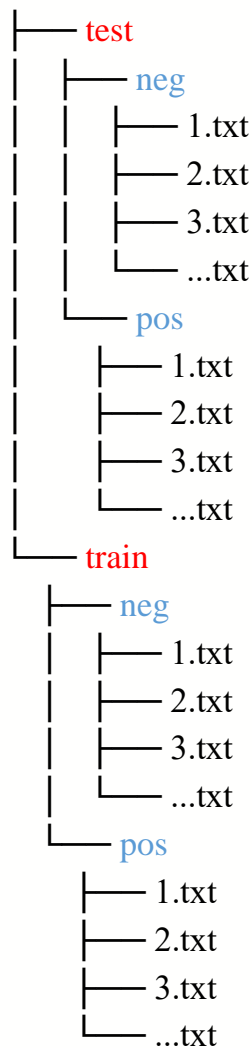
1.2.1. Thông tin bộ dữ liệu bình luận bằng tiếng Việt

- Bộ dữ liệu sử dụng là các bình luận được lấy từ trang web foody.vn do streetcodevn.com thực hiện.

(Bộ dữ liệu: <https://streetcodevn.com/blog/dataset>).

- Thông tin bộ dữ liệu bao gồm 50.000 bình luận đã được xử lý tách từ:
 - + Số lượng mẫu tập train là 30.000 bình luận.
 - + Số lượng mẫu tập validate là 10.000 bình luận.
 - + Số lượng mẫu tập test là 10.000 bình luận.

- Cấu trúc dữ liệu thư mục data_train:



1.2.2. Chuẩn hóa dữ liệu Tiếng Việt

Chuẩn hóa dữ liệu trong Tiếng Việt bước đầu giống như xử ở Tiếng Anh, là dữ liệu đầu vào phải được xóa các kí tự đặc biệt, xóa các dấu cách ở đầu và cuối câu, đảm bảo giữa.

Tuy nhiên, đặc điểm từ ngữ Tiếng Việt có cấu trúc phức tạp, từ ngữ bao gồm từ đơn và từ phức (kết hợp 2 từ trở lên thành 1 từ có nghĩa) nên việc phân biệt *từ đơn* hay *từ phức* trở thành một bài toán lớn. Do đó đảm bảo giữa các từ chỉ có một dấu khoảng trắng (giữa các từ cấu thành *từ phức* có dấu gạch chân “_”) trở nên khó khăn.

Ý tưởng xử lý:

- Ghép cục đại: Đặt các từ vào câu sao cho phủ hết được câu đó, thỏa mãn một số heuristic nhất định. Phương pháp này các ưu điểm là rất nhanh, nhưng có rất nhiều hạn chế, ví dụ như độ chính xác thấp, không xử lý được những từ không có trong từ điển.

- Luật: Xây dựng tập luật bằng tay hoặc tự động để phân biệt các cách kết hợp được phép và không được phép.

- Đồ thị hoá: Xây dựng một đồ thị biểu diễn câu và giải bài toán tìm đường đi ngắn nhất trên đồ thị.

- Gán nhãn: Coi như bài toán gán nhãn chuỗi. Cách này được sử dụng trong JVNSegmenter, Đông du.

- Dùng mô hình ngôn ngữ: Cho trước một số cách tách từ của toàn bộ câu, một mô hình ngôn ngữ có thể đánh giá được cách nào có khả năng cao hơn. Đây là cách tiếp cận của vnTokenizer.

Để tránh tập trung quá nhiều vào một bài toán khác, bộ dữ liệu được chọn dùng ở trên là bộ dữ liệu Tiếng Việt đã được xử lý tách từ.

2. Xử lý tách từ

- Các câu trong tập dữ liệu được chuẩn hóa sẽ được tách thành cách từ riêng biệt nhờ dấu khoảng cách giữa mỗi từ.

3. Xây dựng bộ từ vựng từ tập các câu train và test

- Mỗi từ trong tập dữ liệu huấn luyện sử dụng sẽ được gán một giá trị số nguyên lớn hơn 0.

- Các số nguyên này không phải được gán một cách ngẫu nhiên mà sẽ theo một nguyên tắc nhất định. Nguyên tắc gán số cho từ dựa trên số lần xuất hiện của từ đó trong tập dữ liệu. Xuất hiện nhiều lần, tức là từ đó quá phổ thông, mức độ quan trọng không cao, cho nên sẽ được gán số nhỏ. Ngược lại, nếu từ đó có số lần xuất hiện thấp thì số được gán sẽ cao.

4. Số hóa các từ trong câu

- Sau khi thực hiện xây dựng bộ từ vựng, chúng ta tiến hành thay thế các từ trong câu thành số được gán trên bộ từ vựng.

II. Đưa các câu đã số hóa vào Embedding

1.Embedding là gì?

- Embedding là một kỹ thuật đưa một vector có số chiều lớn, thường ở dạng thưa, về một vector có số chiều nhỏ, thường ở dạng dày đặc. Phương pháp này đặc biệt hữu ích với những đặc trưng hạng mục có số phần tử lớn ở đó phương pháp chủ yếu để biểu diễn mỗi giá trị thường là một vector dạng one-hot.

2. Ý nghĩa của việc đưa dữ liệu qua Embedding

- Việc đưa dữ liệu số hóa vào embedding là để các từ có ý nghĩa tương tự nhau nằm gần nhau trong không gian embedding. Khi đó mô hình LSTM sẽ học được nhiều mối quan hệ hơn.

III. Mô hình LSTM – Long Short Term Memory networks

- Đưa dữ liệu là các vector đã được embedding xử lý vào mô hình LSTM để học các mối quan hệ giữa các từ.

- Đầu ra của LSTM là một vector chứa thông tin đã học là các mối quan hệ giữa các từ trong câu, hay giữa các tế bào trong mô hình LSTM.

IV. Lớp dự đoán thông thường Dense

1. Dense layer là gì?

- Dense layer hay Fully-connected layer là một lớp cổ điển trong mạng nơ ron nhân tạo. Mỗi nơ ron nhận đầu vào từ tất cả nơ ron lớp trước đó.

- Bên trong lớp Dense sử dụng các phép nhân ma trận để thay đổi kích thước một vector.

2. Tác dụng của lớp Dense trong mô hình

- Đầu ra của LSTM là một vector khá phức tạp, nên việc dự đoán nhãn cho một câu là chưa thể. Do đó, lớp dc thêm vào để biến vector đầu ra của LSTM thành một giá trị duy nhất, nó chính là xác suất để một câu là tích cực. Tức là giá trị đầu ra của lớp Dense trong mô hình nằm trong khoảng $[0,1]$:

+ ≥ 0.5 : Tích cực.

+ < 0.5 : Tiêu cực.

CHƯƠNG III: THỰC NGHIỆM MÔ HÌNH LSTM TRONG BÀI TOÁN PHÂN LOẠI CẢM XÚC

I. Với bộ dữ liệu Tiếng Anh



1. Huấn luyện mô hình




- Thêm các thư viện cần thiết:

```
import tensorflow_datasets as tfds
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Embedding
from tensorflow.keras.models import Sequential
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

- Tiến hành tải dữ liệu thư viện **tensorflow_datasets** của **Tensorflow** lưu dưới tên **"imdb"**.

```
#Tiến hành tải dữ liệu về dưới tên "imdb" và thông tin dữ liệu "info"
imdb, info = tfds.load('imdb_reviews', with_info=True, as_supervised=True)
```

```
Downloading and preparing dataset imdb_reviews/plain_text/1.0.0 (download: 80.23 MiB, generated: Unknown size, total: 80.23 MiB)
DI Completed.... 100%  1/1 [00:06<00:00, 6.60s/ uri]
DI Size.... 100%  80/80 [00:06<00:00, 17.93 MiB/s]

Shuffling and writing examples to /root/tensorflow_datasets/imdb_reviews/plain_text/1.0.0.incompleteWQXIA5/imdb_reviews-train.tfr
100%  24999/25000 [00:00<00:00, 121463.97 examples/s]
Shuffling and writing examples to /root/tensorflow_datasets/imdb_reviews/plain_text/1.0.0.incompleteWQXIA5/imdb_reviews-test.tfr
100%  24999/25000 [00:00<00:00, 122750.19 examples/s]
Shuffling and writing examples to /root/tensorflow_datasets/imdb_reviews/plain_text/1.0.0.incompleteWQXIA5/imdb_reviews-unsupervi
100%  49999/50000 [00:00<00:00, 140107.10 examples/s]
```

- Lấy ra tập train và test, sau đó duyệt qua từng phần tử của nó lưu vào các mảng **train_sentences[], test_sentences[]** và **train_labels[], test_labels[]**.

```
train_data, test_data = imdb['train'], imdb['test']

train_sentences = []
test_sentences = []
train_labels = []
test_labels = []

for s, l in train_data:
    train_sentences.append(str(s.numpy()))
    train_labels.append(l.numpy())

for s, l in test_data:
    test_sentences.append(str(s.numpy()))
    test_labels.append(l.numpy())
```

- Ép kiểu các tập nhãn **train_labels[]** và **test_labels[]** sang kiểu numpy để đưa vào mô hình.

```
train_labels = np.array(train_labels)
test_labels = np.array(test_labels)
```

- Thiết lập các hằng số **vocab_size, embedding_dim, max_length**.

```
vocab_size = 10000 #số từ vựng
embedding_dim = 64 #Số chiều embedding
max_length = 140 #Số lượng từ tối đa của một câu
```

- Thiết lập các thông số cho bộ từ vựng và tiến hành xây dựng bộ từ điển từ tập tập các câu **train_sentences[]** và **test_sentences[]** bằng thư viện **Tokenizer**.

```
tokenizer = Tokenizer(num_words=vocab_size, oov_token="<OOV>")
tokenizer.fit_on_texts(train_sentences+test_sentences)
```

- Tiến hành số hóa các từ bằng thư đã xây dựng.

- Sử dụng hàm **pad_sequences** trong thư viện **Keras** cắt bỏ các từ cuối câu nếu số từ trong câu lớn hơn 140, thêm các phần tử 0 vào cuối câu nếu câu chưa đủ 140 từ và lưu vào các tập **padded_train_sequences, padded_test_sequences**.

```
train_sequences = tokenizer.texts_to_sequences(train_sentences)
padded_train_sequences = pad_sequences(train_sequences, maxlen=max_length, truncating='post', padding='post')
test_sequences = tokenizer.texts_to_sequences(test_sentences)
padded_test_sequences = pad_sequences(test_sequences, maxlen=max_length, truncating='post', padding='post')
```


- Tạo mô hình và thêm các lớp cần thiết.

```
model = Sequential()
model.add(Embedding(vocab_size, embedding_dim, input_length=max_length))
model.add(LSTM(32))
model.add(Dense(1, activation='sigmoid'))
```

- Tiến hành xây dựng mô hình.

```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
```

- Chi tiết mô hình lúc này.

```
model.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 140, 64)	640000
lstm (LSTM)	(None, 32)	12416
dense (Dense)	(None, 1)	33

```
=====
Total params: 652,449
Trainable params: 652,449
Non-trainable params: 0
=====
```

- Thực hiện huấn luyện mô hình với các tập dữ liệu **padded_train_sequences**, **train_labels[]**.

```
model.fit(padded_train_sequences, train_labels, epochs=10)
```

```
Epoch 1/10
782/782 [=====] - 26s 26ms/step - loss: 0.4865 - acc: 0.7673
Epoch 2/10
782/782 [=====] - 20s 26ms/step - loss: 0.3203 - acc: 0.8772
Epoch 3/10
782/782 [=====] - 20s 26ms/step - loss: 0.2492 - acc: 0.9097
Epoch 4/10
782/782 [=====] - 20s 26ms/step - loss: 0.2112 - acc: 0.9248
Epoch 5/10
782/782 [=====] - 20s 26ms/step - loss: 0.1730 - acc: 0.9409
Epoch 6/10
782/782 [=====] - 20s 26ms/step - loss: 0.1580 - acc: 0.9467
Epoch 7/10
782/782 [=====] - 20s 26ms/step - loss: 0.1339 - acc: 0.9554
Epoch 8/10
782/782 [=====] - 20s 26ms/step - loss: 0.0860 - acc: 0.9734
Epoch 9/10
782/782 [=====] - 20s 26ms/step - loss: 0.1024 - acc: 0.9680
Epoch 10/10
782/782 [=====] - 20s 26ms/step - loss: 0.0801 - acc: 0.9756
<keras.callbacks.History at 0x7f467c1a8c50>
```

2. Đánh giá mô hình với bộ dữ liệu Tiếng Anh

2.1. Sử dụng mô hình đánh giá Confusion matrix

- Chạy mô hình dự đoán cho tập **padded_test_sequences** và lưu vào **pred**, duyệt qua từng phần tử của kết quả dự đoán, biến kết quả dự đoán thành các nhãn 0 và 1 và lưu vào tập **pred_label**. Sau đó ép kiểu tập nhãn dự đoán **pred_label** về numpy.

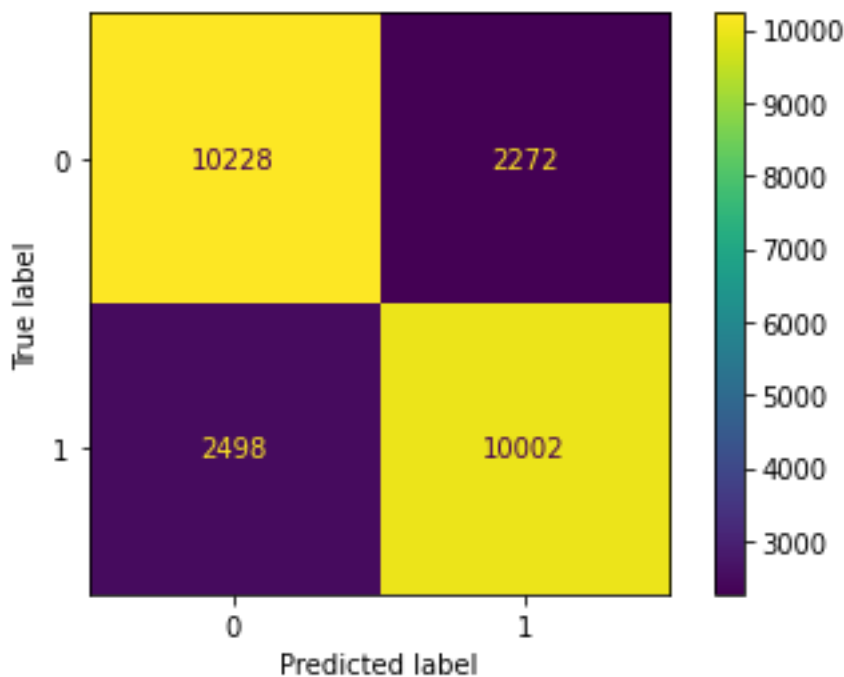
```
pred = model.predict(padded_test_sequences)

pred_labels = []

for i in pred:
    if float(i)>=0.5:
        pred_labels.append(1)
    else:
        pred_labels.append(0)
pred_labels = np.array(pred_labels)
```

- Tiến hành tạo Confusion Matrix **cf_matrix** với đầu vào là nhãn thực tế **test_labels** và nhãn dự đoán **pred_labels**. Tạo mô hình đánh giá trực quan với **ConfusionMatrixDisplay** lưu mô hình trực quan dưới tên **disp**. Đưa mô hình trực quan vào **Matplotlib** và vẽ Confusion Matrix.

```
cf_matrix=confusion_matrix(test_labels,pred_label)
disp = ConfusionMatrixDisplay(confusion_matrix=cf_matrix)
disp.plot()
plt.title("Confusion Matrix\n")
plt.show()
```



Hình 8: Confusion Matrix cho mô hình LSTM với Tiếng Anh

- Công thức tính độ chính xác của mô hình dựa vào biểu đồ Confusion Matrix.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} * 100 \quad (1)$$

- Trong đó

- + TP là số phần tử tích cực mà mô hình dự đoán đúng.
- + FP là số phần tử tích cực mà mô hình dự đoán sai.
- + TN là số phần tử tiêu cực mà mô hình dự đoán đúng.
- + FN là số phần tử tiêu cực mà mô hình dự đoán sai.

- Tiến hành tính độ chính xác cho mô hình.

$$\text{Accuracy} = \frac{10002 + 10228}{10002 + 10228 + 2498 + 2272} * 100 = 80,92\%$$

2.2. Đánh giá với một câu mới

- Xây dựng một câu mới lưu vào **test_sen**, sau đó thực hiện số hóa câu mới bằng bộ từ vựng đã xây dựng lưu **test_seq**. Đưa câu về đúng dạng 140 từ bằng cách cách bỏ hoặc thêm phần tử 0 vào cuối và lưu vào **padded_test_seq**.

```
test_sen = ["The movie is very bad"]
test_seq = tokenizer.texts_to_sequences(test_sen)
padded_test_seq = pad_sequences(test_seq, maxlen=max_length, truncating="post",
padding="post")
```

- Tiến hành dự nhãn cho câu mới.

```
model.predict(padded_test_seq)
```

```
array([[0.05062872]], dtype=float32)
```

- Đến đây mô hình đã có thể dự đoán nhãn cho một câu mới.

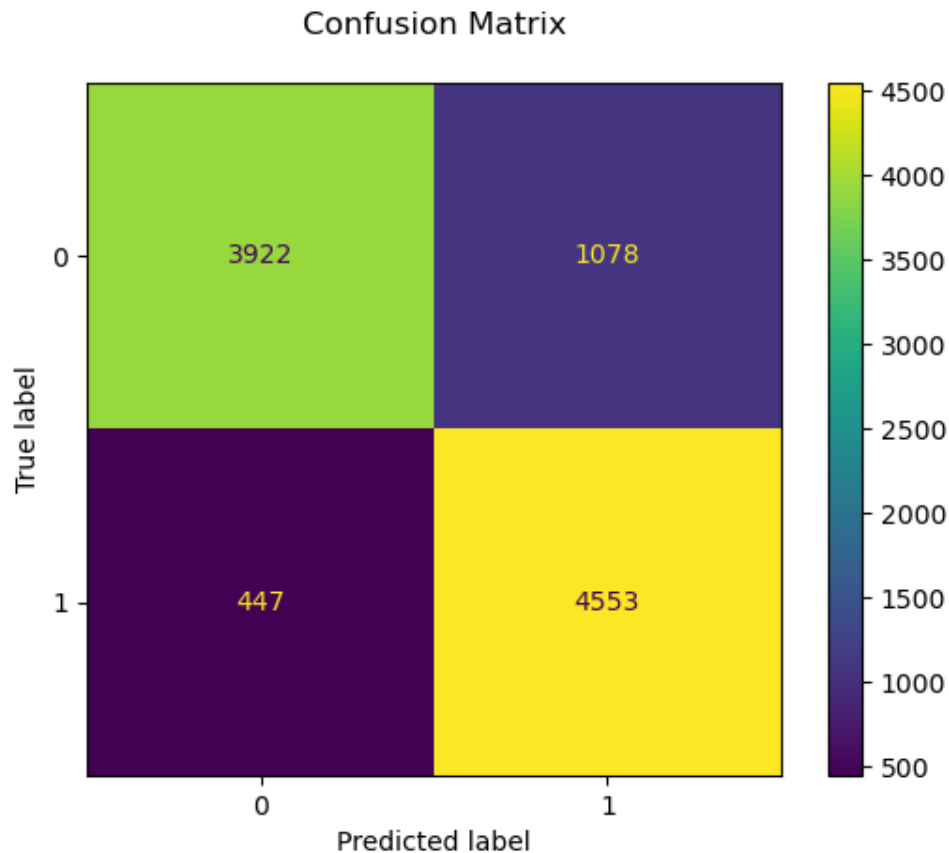
II. Với bộ dữ liệu Tiếng Việt

1. Huấn luyện mô hình

- Do tập dữ liệu dùng để huấn luyện là tập dữ liệu tiếng Việt đã được chuẩn hóa (đã được xử lý tách từ) nên các bước huấn luyện được thực hiện tương tự đối với mô hình phân loại cảm xúc bằng LSTM đối với dữ liệu tiếng Anh.

2. Đánh giá mô hình

2.1. Sử dụng mô hình đánh giá Confusion Matrix



- Áp dụng công thức (1) ta thu được tỷ lệ chính xác của mô như sau:

$$\text{Accuracy} = \frac{4553 + 3922}{4553 + 3922 + 447 + 1078} * 100 = 84,75\%$$

2.2. Đánh giá mô hình với câu mới

- Xây dựng câu mới với nội dung “Món ăn này rất ngon và sạch sẽ”.

```
test_sen = ["Món ăn này rất ngon và sạch sẽ"]
test_seq = tokenizer.texts_to_sequences(test_sen)
padded_test_seq = pad_sequences(test_seq, maxlen=max_length, truncating="post",
padding="post")
```

- Tiến hành dự đoán.

```
model.predict(padded_test_seq)
```

```
array([[0.76995434]], dtype=float32)
```

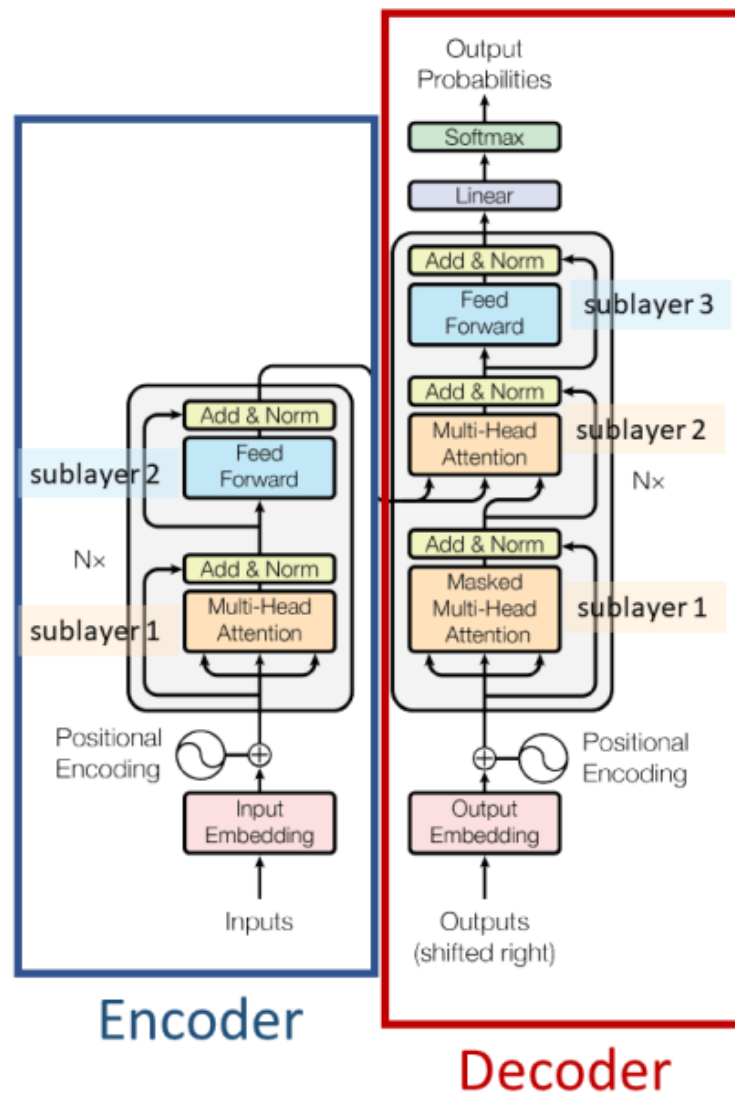
- Đã dự đoán nhãn chính xác nhãn cho câu mới.

PHẦN II: MÔ HÌNH TRANSFORMERS

CHƯƠNG I: TÌM HIỂU MÔ HÌNH TRANSFORMER TRONG BÀI TOÁN PHÂN LOẠI CẢM XÚC

I. Mô hình Transformer là gì?

- Mô hình Transformer là một mô hình học sâu sử dụng chủ yếu trong việc xử lý ngôn ngữ tự nhiên. Cấu trúc của Transformer gồm 2 phần chính là Encoder và Decoder gần giống với RNN. Tuy nhiên điểm làm Transformer trở nên nổi trội chính là các input được đẩy vào cùng một lúc (xử lý song song) nên tốc độ huấn luyện mô hình sẽ nhanh hơn – điều này đã khắc phục được nhược điểm của kiến trúc tuần tự trong mô hình RNN.



Hình 9: Cấu trúc của mô hình Transformer
(Nguồn: <https://viblo.asia/>)

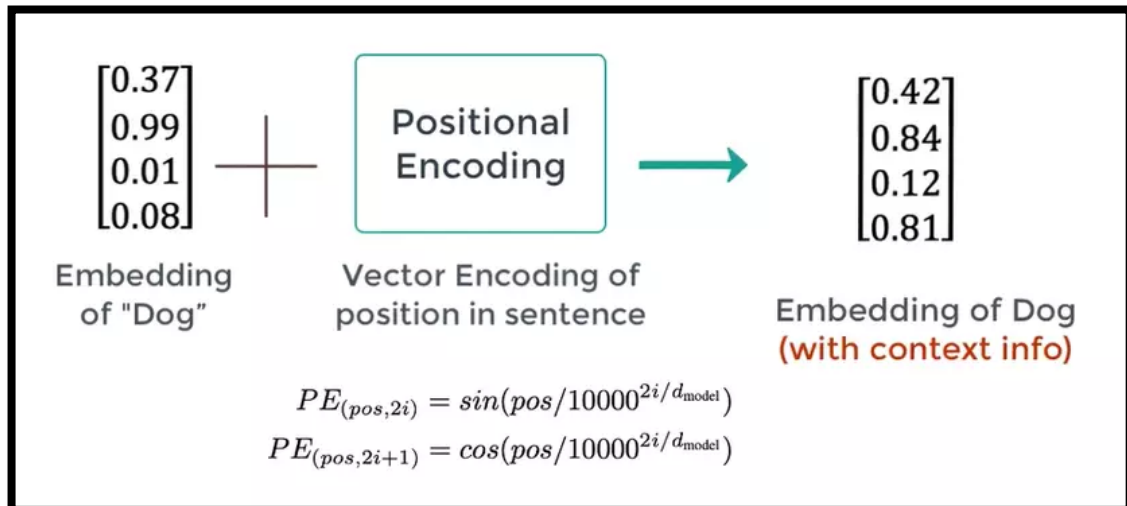
II. Transformer trong bài toán phân loại cảm xúc

- Đối với bài toán phân loại cảm xúc, chúng ta chỉ sử dụng phần Encoder của mô hình Transformer thay vì sử dụng toàn bộ mô hình. Sau đó dùng output của Encoder đưa qua một lớp Dense để tiến hành phân loại cảm xúc.

- Chi tiết mỗi Encoder của mô hình Transformer gồm

1. Positional Encoding

- Cùng một từ khi nằm ở vị trí khác nhau sẽ mang ý nghĩa khác nhau. Đó là lý do Transformers có thêm một phần Positional Encoding để biểu diễn nghĩa của từ ở các vị trí khác nhau trong câu.



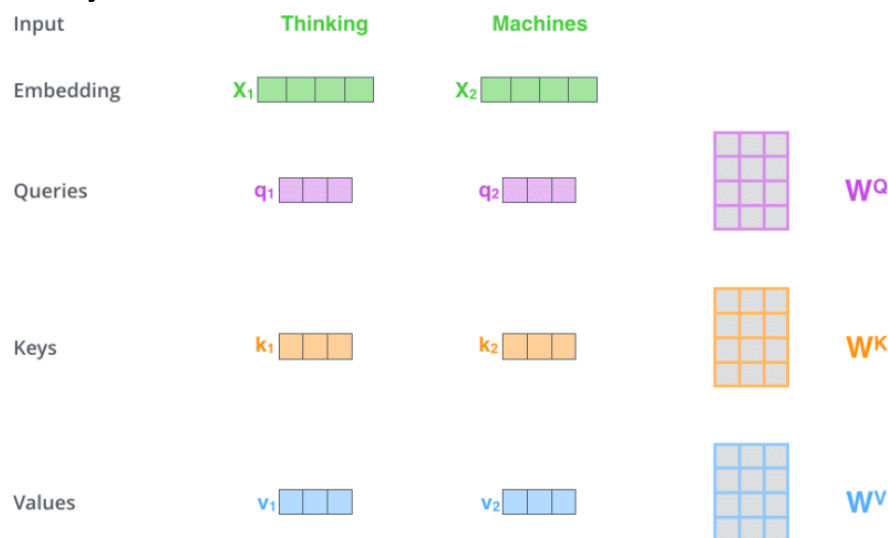
Hình 10: Positional Encoding

(Nguồn: <https://images.viblo.asia/>)

- Trong đó pos là vị trí của từ trong câu, PE là giá trị phần tử thứ i trong embeddings có độ dài d_{model} . Sau đó ta cộng PE vector và $Embedding$ vector.

2. Self-Attention

- Với mỗi từ, ta sẽ có 3 vector Query, Key và Value. Các vector này được tạo nên bởi phép nhân ma trận giữa vector đầu vào và 3 ma trận trọng số chúng ta sử dụng trong quá trình huấn luyện.



Hình 11: 3 vector Q, K, V của mỗi từ

(Nguồn: <https://images.viblo.asia/>)

- Công thức để tính Attention của một từ:

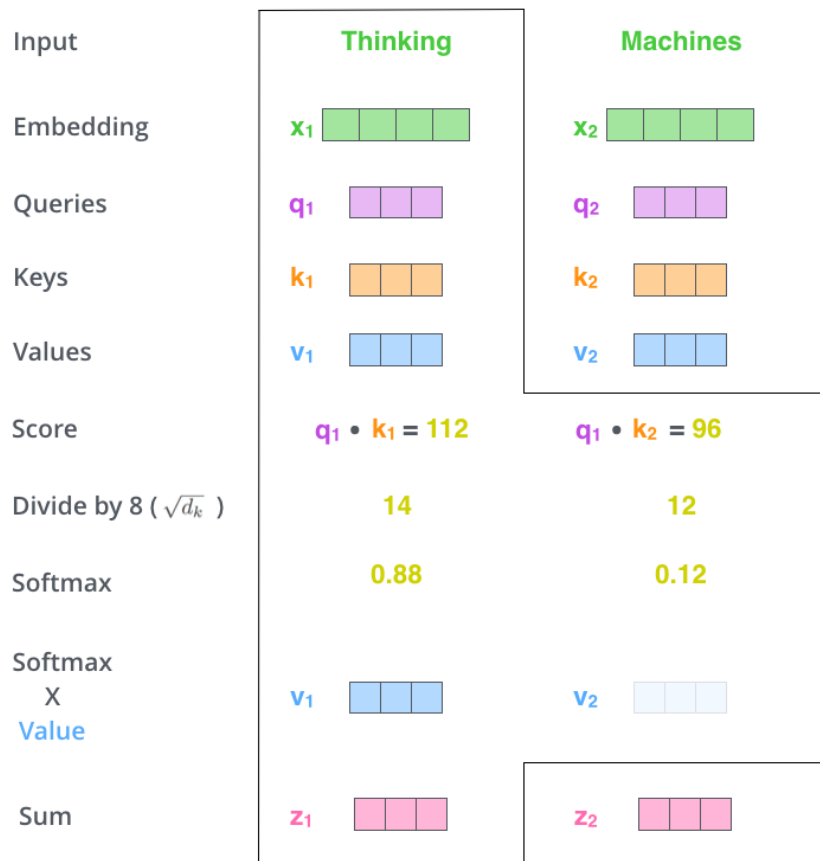
$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_K}}\right)V$$

- Đầu tiên ta sẽ tính mức độ liên quan của từ đang xét với các từ trong câu bằng cách nhân vector Q của từ hiện tại với vector K của từng từ trong câu. Điểm càng lớn thì mức độ liên quan càng lớn và ngược lại.

- Bước tiếp theo là chuẩn hóa điểm bằng cách chia điểm cho căn bậc 2 số chiều của vector K (trong hình dưới lấy $d_K=64$). Điều này giúp cho độ dốc trở nên ổn định hơn.

- Tiếp theo, giá trị này được truyền qua hàm softmax để đảm bảo các giá trị điểm đều dương và có tổng không vượt quá 1

- Kế tiếp, ta nhân giá trị đã qua hàm softmax với vector V để loại bỏ những từ không cần thiết (xác suất nhỏ) và giữ lại những từ quan trọng (xác suất lớn). Sau đó ta cộng các giá trị này lại với nhau tạo nên véc tơ Z duy nhất cho một từ (đây chính là Attention của một từ).



Hình 12: Các bước tính Attention của một từ
(Nguồn: <https://images.viblo.asia/>)

3. Multi-head Attention

- Mỗi từ sẽ chú ý đến từ khác trong câu mà từ đó liên quan đến nó nhất, các từ khác nhau chúng sẽ chú ý đến từ khác nhau. Để biểu diễn mối quan hệ một cách khái quát nhất. Thay vì sử dụng một Attention biểu thị mối quan hệ giữa một với các từ trong câu ta tính Attention cho từng từ trong câu và nối chúng lại với nhau. Sau đó nhân với trọng số W^O để tạo ra một Attention duy nhất cho mỗi câu.

1) Concatenate all the attention heads



2) Multiply with a weight matrix W^O that was trained jointly with the model

x



3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN



TÀI LIỆU THAM KHẢO

- [1][<https://viblo.asia/p/transformers-nguoi-may-bien-hinh-bien-doi-the-gioi-nlp-924IJPOXKPM>]
- [2][[https://vi.wikipedia.org/wiki/Transformer_\(m%C3%B4_h%C3%ACnh_h%E1%BB%8Dc_m%C3%A1y\)](https://vi.wikipedia.org/wiki/Transformer_(m%C3%B4_h%C3%ACnh_h%E1%BB%8Dc_m%C3%A1y))]
- [3][https://www.tensorflow.org/text/tutorials/classify_text_with_bert]
- [4][<https://colah.github.io/posts/2015-08-Understanding-LSTMs/>]
- [5][<https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf>]
- [6][<https://trituenhantao.io/kien-thuc/transformer-hoat-dong-nhu-the-nao/>]