BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC CẦN THƠ**

**KHOA CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG**

****

**LUẬN VĂN TỐT NGHIỆP ĐẠI HỌC**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

**Đề tài**

**XÂY DỰNG HỆ THỐNG TRẢ LỜI TỰ ĐỘNG**

**Sinh viên: Ngụy Hữu Lộc**

**Mã số: B1706606**

**Khóa: K43**

**Cần Thơ, 02/2021**

BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC CẦN THƠ**

**KHOA CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG**

**LUẬN VĂN TỐT NGHIỆP ĐẠI HỌC**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

**Đề tài**

**XÂY DỰNG HỆ THỐNG TRẢ LỜI TỰ ĐỘNG**

**Người hướng dẫn Sinh viên thực hiện**

**TS. Lâm Nhựt Khang Ngụy Hữu Lộc**

**Mã số: B1706606**

**Khóa: K43**

***Cần Thơ, 02/2021***

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

Cần Thơ, ngày …tháng … năm …

Giáo viên hướng dẫn

TS. Lâm Nhựt Khang

LỜI CAM ĐOAN

.

Cần Thơ, ngày ….. tháng ……… năm 2021

LỜI CẢM ƠN

Cần Thơ, ngày ….. tháng …… năm 2021

Sinh viên thực hiện

MỤC LỤC

[NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN i](#_Toc65322773)

[LỜI CAM ĐOAN ii](#_Toc65322774)

[LỜI CẢM ƠN iii](#_Toc65322775)

[MỤC LỤC iv](#_Toc65322776)

[DANH MỤC HÌNH ẢNH vi](#_Toc65322777)

[DANH MỤC BIỂU BẢNG vii](#_Toc65322778)

[DANH MỤC TỪ VIẾT TẮT viii](#_Toc65322779)

[TÓM TẮT ix](#_Toc65322780)

[ABSTRACT x](#_Toc65322781)

[CHƯƠNG 1. GIỚI THIỆU TỔNG QUAN 1](#_Toc65322782)

[1.1. Tổng quan 1](#_Toc65322783)

[1.2. Nghiên cứu liên quan 1](#_Toc65322784)

[1.3. Mục tiêu đề tài 1](#_Toc65322785)

[1.4. Đối tượng và phạm vi nghiên cứu 1](#_Toc65322786)

[1.5. Phương pháp nghiên cứu 1](#_Toc65322787)

[1.6. Nội dung nghiên cứu 1](#_Toc65322788)

[1.7. Bố cục 2](#_Toc65322789)

[CHƯƠNG 2. CƠ SỞ LÝ THUYẾT 3](#_Toc65322790)

[2.1. Mạng neural nhân tạo 3](#_Toc65322791)

[2.1.1. Kiến trúc mạng neural nhân tạo 3](#_Toc65322792)

[2.1.2. Quá trình xử lý thông tin của một mạng neural 4](#_Toc65322793)

[2.1.3. Recurrent Neural Network 5](#_Toc65322794)

[2.1.4. Long Short Term Memory network 6](#_Toc65322795)

[2.2. Transformer 8](#_Toc65322796)

[2.2.1 Tổng quan 8](#_Toc65322797)

[2.2.2 Encoder 10](#_Toc65322798)

[2.2.3 Decoder 15](#_Toc65322799)

[2.3. Lý thuyết 3 15](#_Toc65322800)

[2.4. Phương pháp đánh giá 15](#_Toc65322801)

[CHƯƠNG 3. PHƯƠNG PHÁP THỰC HIỆN 16](#_Toc65322802)

[CHƯƠNG 4. THỰC NGHIỆM 18](#_Toc65322803)

[4.1. Xây dựng tập dữ liệu 18](#_Toc65322804)

[4.2. Tiền xử lý dữ liệu 18](#_Toc65322805)

[4.3. Huấn luyện mô hình 18](#_Toc65322806)

[4.4. Kiểm thử và điều chỉnh 18](#_Toc65322807)

[4.5. Đánh giá độ chính xác 18](#_Toc65322808)

[4.6. Thảo luận kết quả thực nghiệm 18](#_Toc65322809)

[CHƯƠNG 5. KẾT LUẬN 19](#_Toc65322810)

[5.1. Kết quả đạt được 19](#_Toc65322811)

[5.2. Hướng phát triển 19](#_Toc65322812)

[TÀI LIỆU THAM KHẢO 20](#_Toc65322813)

[PHỤ LỤC 21](#_Toc65322814)

DANH MỤC HÌNH ẢNH

DANH MỤC BIỂU BẢNG

DANH MỤC TỪ VIẾT TẮT

| **Từ viết tắt** | **Mô tả** |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

TÓM TẮT

ABSTRACT

1. GIỚI THIỆU TỔNG QUAN

Chatbot ngày càng xuất hiện nhiều hơn trong cuộc sống của con người, chúng dần đã trở thành một thứ không thể thiếu trong bất kỳ hệ thống nào. Ngày càng có vai trò góp phần thay đổi cuộc sống của con người hiện đại hơn.

* 1. Tổng quan

Ngày nay cùng với sự phát triển của các nền tảng trí tuệ nhân tạo, các ứng dụng trí tuệ nhân tạo ngày cũng xuất hiện nhiều hơn trong cuộc sống. Nếu thời gian trước đây để có thể được hỗ trợ về một sản phẩm hay một vấn đề, nhưng những năm gần đây, sự xuất hiện của nhân viên tư vấn người thật gần như đã không còn nhiều mà thay vào đó là nền tảng chatbot được đưa vào các website để có tự động trả lời các câu hỏi cũng như đưa ra các thông tin mà khách hàng mong muốn không khác gì sự xuất hiện của một nhân viên tư vấn là người thật.

Công cụ trò truyện trực tiếp ngụ ý rằng bạn đang nói chuyện với một con người, mặc dù các bot hoàn toàn được robot hóa. Được biết đến với nhiều tên gọi khác nhau như là bot đàm thoại, chat AI, trợ lý ảo AI,… Chatbots ngày càng trở nên phổ biến trong cuộc sống ngày nay.

* 1. Nghiên cứu liên quan

//Trình bày các nghiên cứu có liên quan. PHẢI sử dụng trích dẫn theo đúng chuẩn IEEE, làm tự động

* 1. Mục tiêu đề tài

Với cơ sở thực tiến, luận văn này sẽ nghiên cứu về các môt hình để xây dựn một hệ thống trả lời tự động, dựa trên mô hình Transformer để huấn luyện tập dữ liệu. Từ đó xây dựng, cài đặt và thử nghiện mô hình đối thoại sử dụng mạng neural để huấn luyện và đào tạo hệ thống trả lời tự đồng.

* 1. Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu mô hình Transformer

Phạm vi nghiên cứu: luận văn sử dụng các framwork Tensoflow để xây dựng mô hình Transformer

* 1. Phương pháp nghiên cứu

Nghiên cứu về mô hình Transformer để đào tạo hệ thống trả lời tự động nhầm đưa ra các câu trả lời chính xác hơn. Phạm vi nghiên cứu trên phương diện ngôn ngữ tiếng Việt.

* 1. Nội dung nghiên cứu
* Tìm hiểu về mô hình Transformer
* Cách thức hoạt động mô hình Transformer
* Vận dụng mô hình Transformer để xây dựng hệ thống trả lời tự động
* Viết báo cáo, đánh giá và so sánh kết quả đạt được.
  1. Bố cục

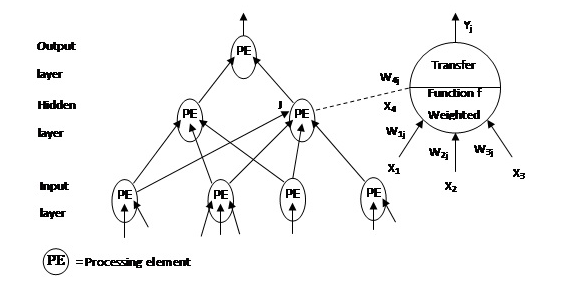
Nội dung của quyển luận văn bao gồm 05 chương:

* Chương 1- Giới thiệu tổng quan: Trình bày tổng quan về chatbot, mục tiêu đề tài, đối tượng, phạm vi, phương pháp và nội dung nghiên cứu.
* Chương 2 - Cơ sở lý thuyết: Nghiên cứu về mạng neural và các phần mở rộng của mạng mạng neural, mô hình Transformer.
* Chương 3- Phương pháp thực hiện: Các phương pháp để xây dựng hệ thống.
* Chương 4- Thực nghiệm Cách thức thực hiện các xử lý mô hình.
* Chương 5- Kết luận: tổng kết kết quả đạt được của nghiên cứu và đề xuất hướng phát triển cho đề tài.

1. CƠ SỞ LÝ THUYẾT

Chương này giới thiệu về mạng neural nhân tạo, cách thức hoạt động của mạng neural, các mô hình mở rộng của mạng neural ( RNN, LSTM). Mô hình Transformer.

* 1. Mạng neural nhân tạo
     1. Kiến trúc mạng neural nhân tạo

Mạng neural nhân tạo ( Artificial Neural Networks[[1]](#footnote-1) – ANN ) là mô hình hóa và mô phỏng sự hoạt động của hệ thần kinh con người. ANN là một mạng phức tạp kết nối các đơn vị tính toán lại với nhau, trong đó mỗi đơn vị tính toán gọi là neural nhân nhân tạo, có thể có nhiều đầu vào nhưng chỉ có một đầu ra duy nhất được minh họa họa Hình 1.

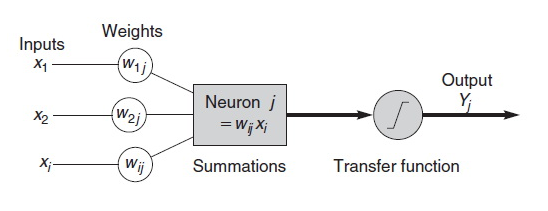
Hình Mô hình mạng neural

( Nguồn : <https://bit.ly/3utuzBA> )

Kiến trúc chung của một ANN gồm có 3 thành phần : Input layer, Hidden layer, Output layer. Trong đó, các các lớp ẩn ( Hidden layer) sẽ nhận dữ liệu từ các neural ở lớp trước ( layer ) và chuyển đổi các input này cho các lớp xử lý phía sau. Trong một ANN có thể có nhiều lớp ẩn.

Các PE ( Processing Elements) của ANN gọi là neural, mỗi neural nhận các dữ liệu vào xử lý chúng và cho ra kết quả duy nhất. Kết quả xử lý của một neural có thể làm input cho các neural khác.

* + 1. Quá trình xử lý thông tin của một mạng neural

Cũng giống như hệ thống mạng thần kinh của con người, mạng neural nhân tạo cũng nhận vào các thông tin và xử lý chúng. Quá trình xử lý của một ANN được biểu diễn như Hình 2.

Hình 2 Quá trình xử lý của mạng neural

( Nguồn : <https://bit.ly/3utuzBA> )

Trong đó :

* Input : mỗi Input tương ứng với một thuộc tính ( attribute) của dữ liệu vào ( patterns).
* Output : kết quả tính toán của một ANN là một giải pháp cho một vấn đề.
* Connection weight ( Trọng số liên kết ) : đây là thành phần rất qua trong với một ANN, nó thể hiện độ quan trọng của dữ liệu đầu vào và đối với quá trình xử lý thông tin ( quá trình chuyển đổi từ layer này sang layer khác ). Quá trình học của ANN thực ra là quá trình điều chỉnh các trọng số (weight) của các input để có được kết quả mong muốn.
* Summation function (hàm tổng): Tính tổng các trọng số của tất cả các input được đưa vào mỗi neural ( phần xử lý PE). Hàm tổng của một neural đối với n input được tính theo công thức sau:

|  |  |
| --- | --- |
|  | (1) |

* Transfer function ( hàm chuyển đổi ): hàm tổng của một neural cho biết khả năng kích hoạt ( activation) của neural đó, còn gọi là kích hoạt bên trong ( internal activation). Các kết quả tính toán của một neural có thể được chuyển đến để làm input cho một neural khác hoặc không.Giả sử có 3 input , , tưởng ứng với 3 weight là , , , mối quan hệ giữ internal activation và kết quả được thể hiện bằng hàm chuyển đổi như sau :

|  |  |
| --- | --- |
| Y = = | ( Với là số thực bất kỳ) |
|  | ( Với là số thực trong đoạn [0;1] |

Việc lựa chọn hàm chuyển đổi có tác động rất lớn đến kết quả của ANN. Hàm chuyển đổi phi tuyến tính được sử dụng phổ biến là sigmoid function:

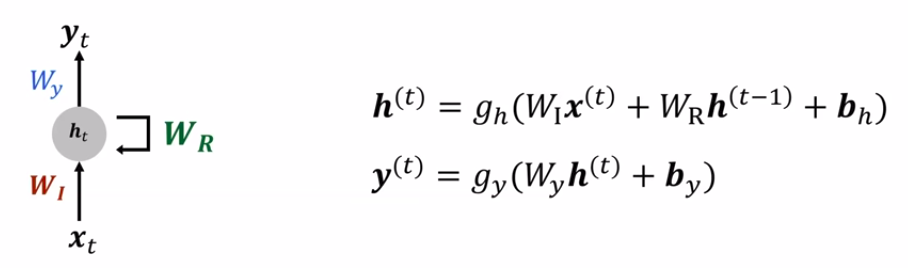
|  |  |
| --- | --- |
|  | (2) |

Trong đó là hàm chuyển đổi và Y là hàm tính tổng.

Kết quả của sigmoid function thuộc khoảng [0;1], nên còn gọi là hàm chuẩn hóa. Kết quả xử lý tại các neural đôi khi rất lớn, vì vậy transfer function được sử dụng để xử lý kết quả này trước khi chuyển đến lớp tiếp theo. Đôi khi thay vì sử dụng transfer function, người ta sử dụng giá trị ngưỡng ( threshold value) để kiếm soát các kết quả của các neural tại một lớp nào đó trước khi chuyển các kết quả này đến các lớp tiếp theo. Nếu kết quả của một neural nào đó nhỏ hơn giá trị ngưỡng thì nó sẽ được chuyển đến lớp tiếp theo.

* + 1. Recurrent Neural Network

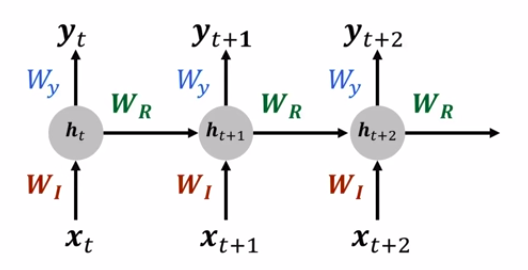
Trong mô hình ANN thông thường (Feed forward netword[[2]](#footnote-2)), chỉ xem các input là dữ liệu độc lập, không có liên kết với nhau. Tuy nhiên, với ngôn ngữ tự nhiên, các từ trong một câu văn lại có liên kết với nhau quyết định nên ý nghĩa của câu văn đó. Chính vì vậy, việc áp dụng một ANN thông thường vào các bài toán xử lý ngôn ngữ tự nhiên thường không đạt được kết quả như mong muốn.

Để có thể khắc phục được nhược điểm trên một mô hình mới được ra đời chính là Recurrent Neural Network[[3]](#footnote-3) – RNN . RNN coi dữ liệu đầu vào là một chuỗi (sequence) liên tục, nối tiếp nhau theo thứ tự thời gian. Ví dụ như một câu văn , được coi là một chuỗi các từ hoặc một chuỗi các ký tự. Tại thời điểm t, với input là ta có kết quả là . Tuy nhiên khác với ANN, lại được sử dụng để làm input để tính kết quả cho thời thời điểm t+1. Điều này cho phép RNN có thể lưu trữ và truyền thông tin đến thời điểm tiếp theo. Mô hình hoạt động của RNN được mô tả như Hình 3 .

Hình 3 Recurrent Neural Network

( Nguồn : <http://bit.ly/3kutCVg> )

Thông thường hàm activation được sử dụng là hàm tanh. Còn hàm chuyển đổi có thể là sigmoid hoặc sofmax tùy theo từng bài toán cự thể.

Để dễ hình dung hơn. Ta chuyển mô hình RNN thành dạng phăng như Hình 4, với là dữ liệu đầu vào tại thời điểm ( timestep) t, t+1, t+2.

Hình 4 Mở rộng của một RNN

( Nguồn : <http://bit.ly/3kutCVg> )

Trong ANN một kết quả cũng có thể trở thành input cho một lớp khác. Tuy nhiên điểm khác nhau giữ ANN và RNN là :

* ANN sử dụng giá trị weight khác nhau cho từng lớp. Ví dụ như kết quả của lớp thứ nhất chỉ là được nhân với , kết quả của lớp thứ hai .
* RNN sử dụng một mạng neural duy nhất ( 1 lớp ) để tính giá trị kết quả cho từng timestep. Do đó các kết quả X sẽ trở thành output sẽ chỉ nhân với cùng một W duy nhất.

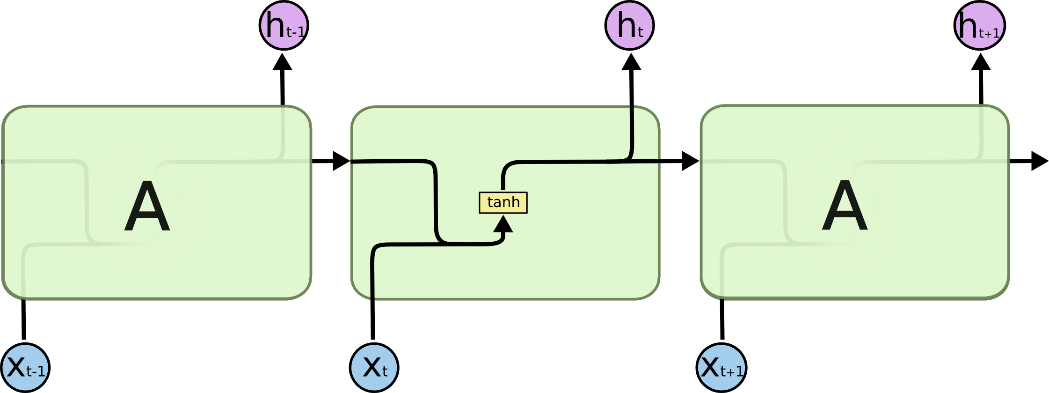
Nói một cách khác, RNN sử dụng một phép tinh duy nhất cho từng phần tử của chuỗi dữ liệu đầu vào, các kết quả sẽ được dùng làm input cho các tính toán kế tiếp.

Tuy nhiên RNN lại gặp phải 2 vấn đề Vanishing Gradient Problem và Exploding Gradient Problem, là những vấn đề gặp phải khi sử dụng gradient để huấn luyện ANN. Vấn đề này do việc lựa chọn các hàm kích hoạt không phù hợp với số lượng các lớp ẩn quá lớn. Đặt biệt là trong các bài toán huấn luyện RNN. Một giải pháp được đưa ra là dùng LSTM sẽ được đề cập bên dưới.

* + 1. Long Short Term Memory network

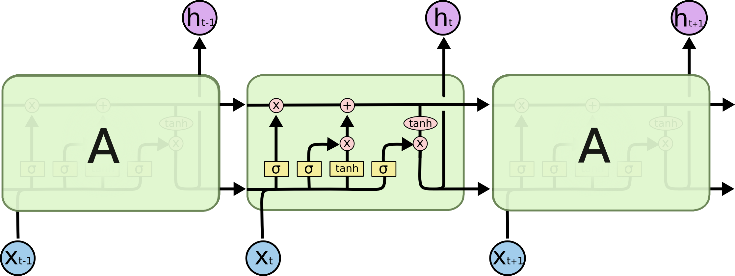
Long Short Term Memory network[[4]](#footnote-4) – LSTM , là một dạng đặc biệt của RNN, nó có khả năng học được các phụ thuộc xa.

Được giới thiệu vào năm 1997 bởi Hochreiter [1] và cộng sự của ông, sau đó được cải tiến và phổ biến trong lĩnh vực xử lý ngôn ngữ tự nhiên. Mạng LSTM có thể giải quyết các vấn đề do việc sử dụng gradient để huấn luyện mạng RNN.

 Với mọi mạng RNN đều có dạng là một chuỗi các mô-đun lặp đi lặp lại của mạng neural. Với mạng RNN chuẩn, các mô-đun này có cấu trúc rất đơn giản, thường là một tầng tanh như Hình 5.

Hình 5 Mô-đun RNN

( Nguồn : <https://dominhhai.github.io/vi/2017/10/what-is-lstm/> )

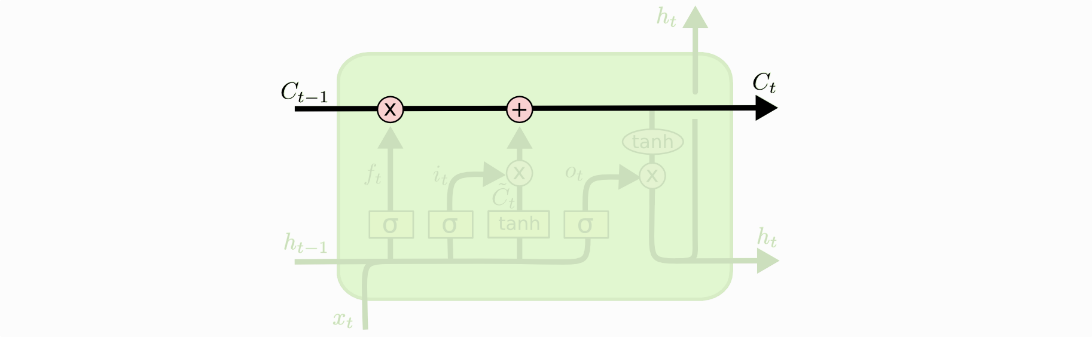
LSTM cũng có kiến trúc dạng chuỗi như vậy, nhưng các mô-đun này có cấu trúc khác với mạng RNN chuẩn. Thay vì chỉ có một tầng mạng neural, LSTM có tới 4 tầng tương ứng với nhau một cách rất đặc biệt được minh họa ở Hình 6.

Hình 6 Mô-đun LSTM

( Nguồn : <https://dominhhai.github.io/vi/2017/10/what-is-lstm/> )

Trong đó :

* Hình chữ nhật là các lớp ẩn của mạng neural.
* Hình tròn buổi diễn toán tử Pointwise.
* Đường kẻ gộp lại với nhau biểu thị phép nối các toán hạng, và.
* Đường rẽ nhánh biểu thị cho sự sao chép tự vị trí này sang vị trí khác.

Phần quan trong nhất trong LSTM chính là các trạng thái tế bào. Trạng thái tế bào giống như bằng truyền. Nó chạy suốt tất cả các mắt xích ( các nút mạng ) và chỉ tương tác tuyến tính đôi chút. Vì vậy mà các thông tin có thể dễ dàng truyền đi thông suốt mà không sợ bị thay đổi, được minh họa như Hình 7.

Hình 7 Trạng thái tế bào

( Nguồn : <https://dominhhai.github.io/vi/2017/10/what-is-lstm/> )

LSTM có khả năng bỏ đi hoặc thêm vào các thông tin cần thiết cho trạng thái tế bào, chúng được điều chỉnh cẩn thận bởi các nhóm được gọi là cổng ( gate ). Các cổng là nơi sàng lọc thông tin qua đó, chúng được kết hợp bởi một tầng mạng sigmoid và một phép nhân.

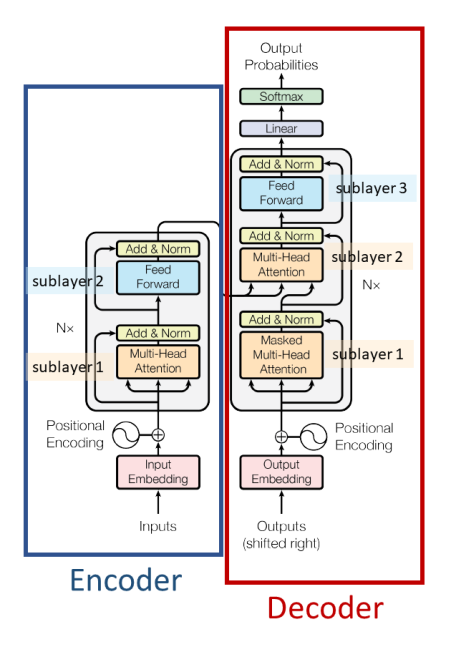
Tầng sigmoid sẽ cho đầu ra là một số trong khoản [0,1], mô tả có bao nhiêu thông tin có thể được thông qua. Khi đầu ra là 0 thì có nghĩa là không cho thông tin nào qua cả, còn khi là 1 thì có nghĩ cho tất cả thoogn tin đi qua nó. Một LSTM gồm có cả 3 cổng như vậy để duy trì và điều hành trạng thái tế bào.

* 1. Transformer

2.2.1 Tổng quan

Transformer là một mô hình học sâu được đề xuất vào năm 2017 bởi Vaswani và các cộng sự của ông [2] . Được sử dụng chủ yếu trong lĩnh vực xử lý ngôn ngữ tự nhiên.

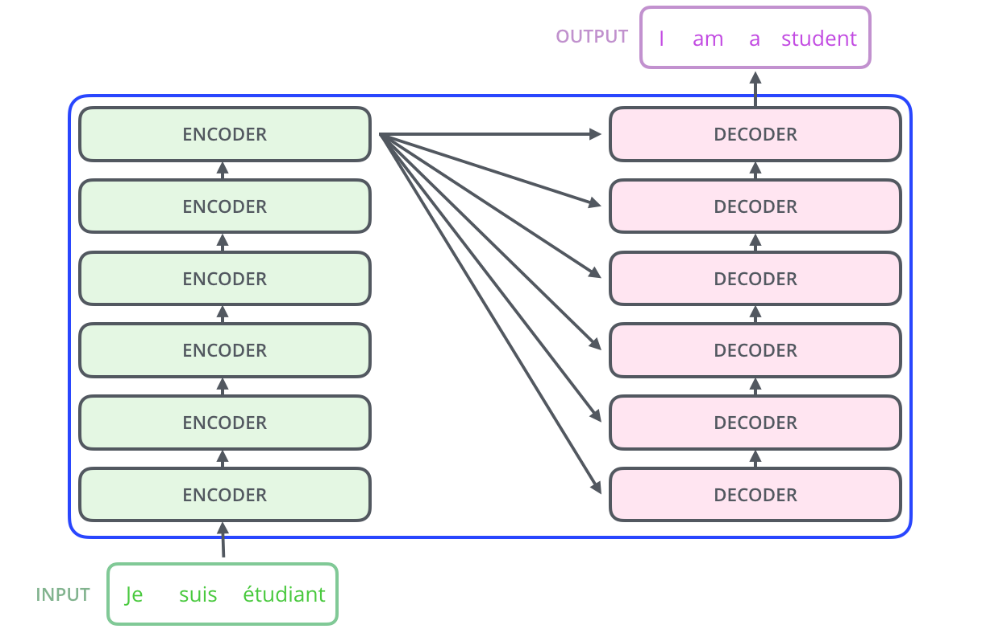
Cũng giống như mô hình RNN hay LSTM, Transformer được thiết kế xử lý dữ liệu tuần tự, chẳng hạn như xử lý ngôn ngữ tự nhiên, cho các công việc như dịch máy hay là tóm tắt văn bản. Tuy nhiên, không giống như RNN, Transformer không yêu cầu dữ liệu tuần tự như được xử lý theo thứ tự, Ví dụ như khi dữ liệu đầu vào là một câu văn, Transformer không cần phải xử lý phần đầu rồi mới xử lý phần cuối. Transformer có thể thực hiện song song nhiều hơn RNN, do đó thời gian đào tạo được giảm đi so với RNN và LSTM.

Kiếm trúc Transformer sử dụng hai phần Encoder và Decoder khá giống với RNN, điểm khác biệt là input được đẩy vào cùng lúc. Với cơ chế này sẽ thay thế cho “recurrent” của RNN. Mô hình cấu trúc của Transformer Hình 8.

Hình 8 Mô hình Transformer

( Nguồn : <https://bit.ly/37PXlmu> )

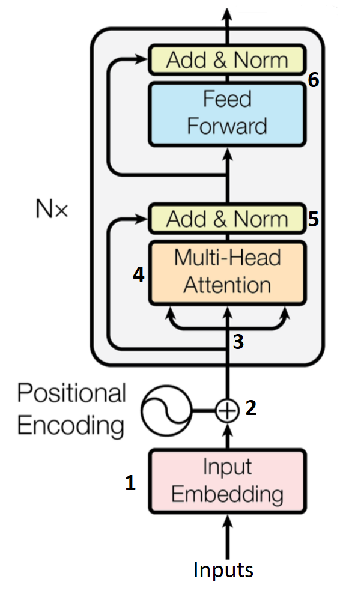
2.2.2 Encoder

Là một khối gồm 6 khối encoder xếp chồng lên nhau. Tất cả các khối đều giống nhau về cấu trúc (nhưng khác nhau về trọng số ) ( Hình 9 )

Hình 9 Cấu tạo cơ bản khối encoder

( Nguồn : <http://jalammar.github.io/illustrated-transformer/> )

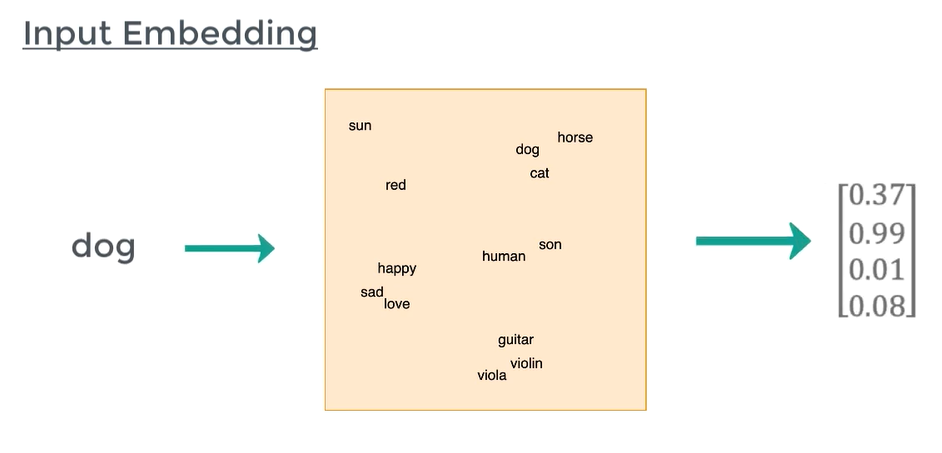
Vai trò của encoder là mã hóa câu đầu vào thành vector ngữ cảnh. Nếu chùng ta có câu dài 5 từ, thì vector ngữ cảnh của nó là Z = (z1, z2, z3, z4, z5 ).

Mô hình chi tiết bên trong của 1 khối Encoder ( Hình 10)

Hình 10 Mô hình chi tiết của khối Encoder

( Nguồn : <http://jalammar.github.io/illustrated-transformer/> )

#### **2.2.2.1 Input Embedding**

Máy tính không thể hiểu được các câu chữ mà chỉ có thể hiểu được các số, vector, ma trận. Vì vậy để máy tính hiểu được, các câu đầu vào phải được biểu diễn dưới dạng vector. Đó cũng là cơ chế chỉnh của lớp Input Embedding trong Encoder. Vì dụ minh họa vector hóa 1 từ như Hình 11.

Hình 11 Ví dụ vector hóa một từ

( Nguồn : <https://bit.ly/37PXlmu> )

#### **2.2.2.2 Position Encoding**

Sau khi đi qua lớp Input Embedding, các câu đã được mã hóa dưới dạng vector, tuy nhiên khi trong câu có nhiều từ lặp lại nhưng khác vị trí lại có ý nghĩa khác nhau. Chính vì điều này, Transformer có thêm một lớp Position Encoding để lưu thông tin về vị trí các từ ở trong câu được tính theo công thức sau :

|  |  |
| --- | --- |
|  | (3) |
|  | (4) |

Trong đó :

* pos: là vị trí từ trong câu.
* PE: là giá trị phần tử thứ i trong Input Embedding có .

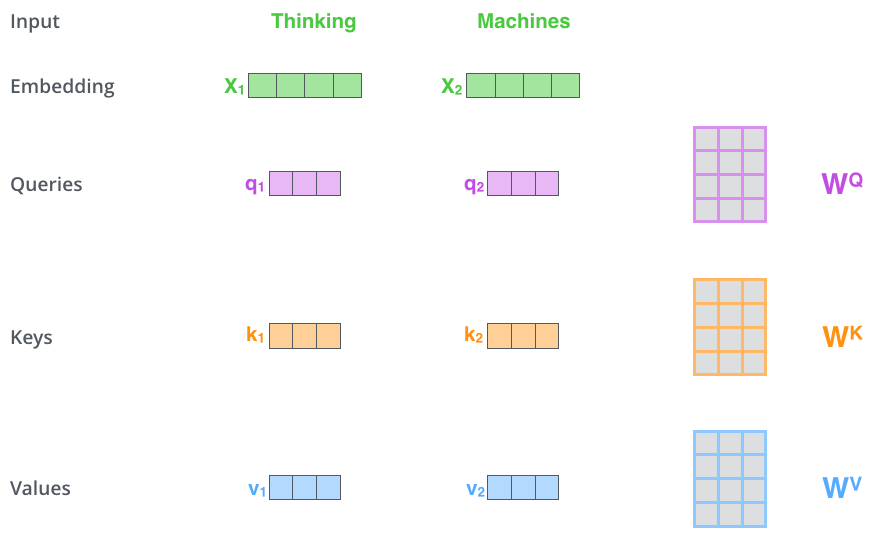
Sau đó cộng các PE vector và Embedding vector lại để được một vector mới lưu thông tin của từ và vị trí của từ trong câu.

#### **2.2.2.3 Self-Attention**

Self-Attention là cơ chế giúp Transformer “hiểu” được sự liên quan giữa các từ trong câu. Ví dụ như câu: “Tôi thích hoa vì nó thật đẹp”. “Nó” ám chỉ điều gì ? Có phải “hoa”. Với con người thì thật đơn giản đê hiểu nhưng với máy tính thì không. Chính vì vậy Self-Attention được sinh ra để giải quyết vấn đề này.

Khi mô hình xử lý từ ( các từ trong câu đầu vào), Self-Attention sẽ xem xét các vị trí khác trong câu đầu vào để tìm ra được sự liên kết và giúp cho mô hình mã hóa tốt hơn.

Cùng tìm hiểu các làm việc của Self-Attention:

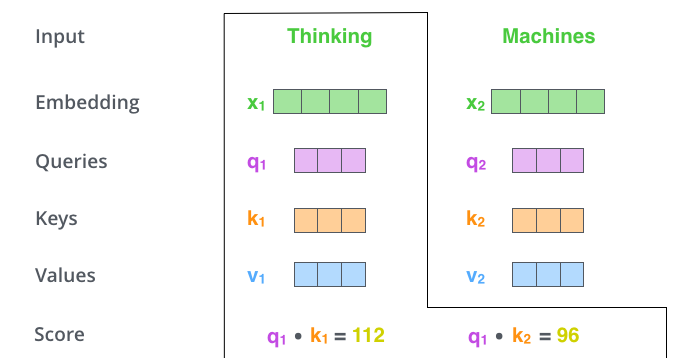
Bước thứ nhất, tạo ra 3 vector, từ mỗi vector đầu vào của bộ mã hóa. Các vector đó lần lượt là vector Q ( Querys), vector (Key) và vector V (Value). Các vector này được tạo ra bằng cách nhân vector đầu vào với ba ma trận ( các ma trận này được sinh ra ngẫu nhiên và thay đổi trong quá trình đào tạo mô hình ) (Hình 12 )

Hình 12 Các vector và ma trận trong Sefl-Attention

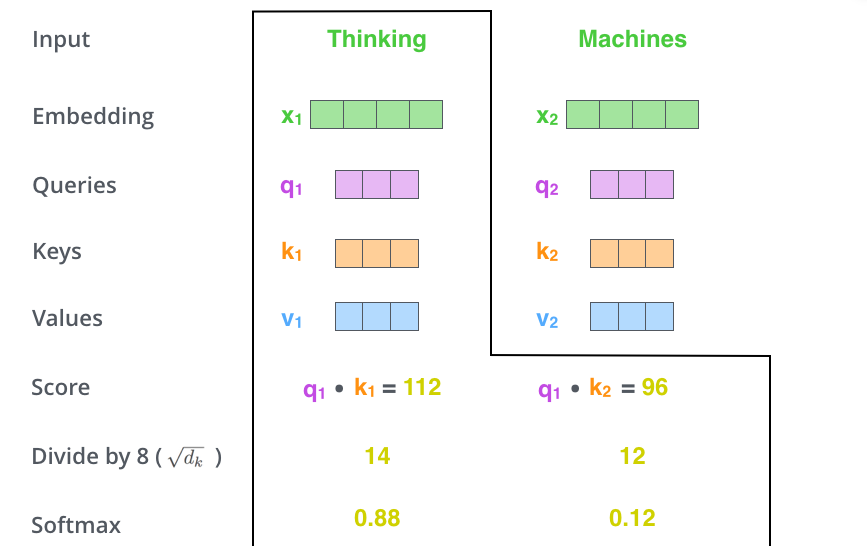
( Nguồn : <http://jalammar.github.io/illustrated-transformer/> )

Lần lượt nhân vector với các ma trận xin ra được vector q1, k1 và v1 ( tương ứng với từ “Thinking”). Tương tự với từ “Machinese”, cũng sẽ có bộ ba vector q2, k2, v2. Tùy theo dộ dài câu mà sinh ra được số bộ vector tương ứng.

Bước thứ hai, lấy vector q1 vừa được tạo nhân lần lượt với từng vector k1 và k2 ( Hình 13 )

Hình Nhân vector q1 với k1 và k2

( Nguồn: <http://jalammar.github.io/illustrated-transformer/> )

Bước thứ ba và thứ tư là chia cho căn bặc 2 số chiều của k1 và k2. Điều này làm cho các gradient ổn định hơn. Sau chuyển chuyển hết cho sofmax. (Hình 14)

Hình 14 Chia score cho số chiều của k1 và k2

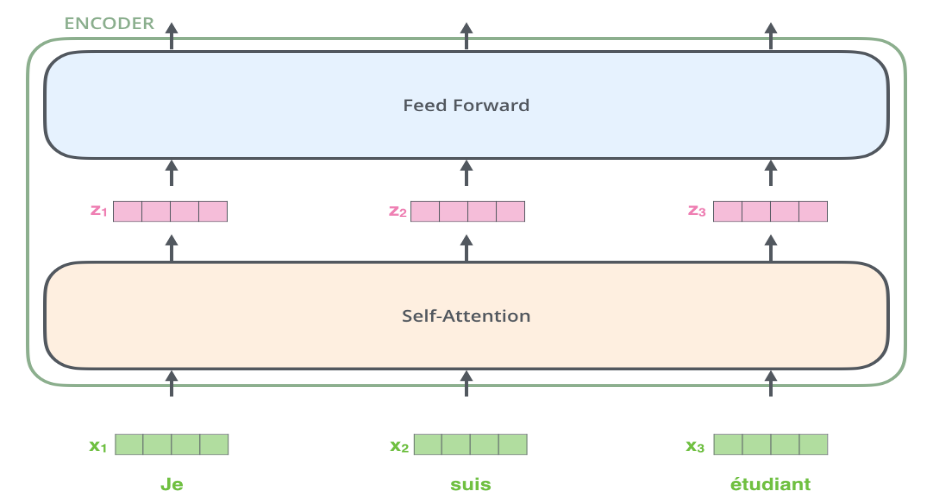
(Nguồn : <http://jalammar.github.io/illustrated-transformer/> )

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

Bước thứ năm, lấy các giá trị sofmax vừa tính được nhân với từng vector v1 và v2. Ở bước này chủ yếu là để giữ nguyên 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 số nhỏ hơn 0.001 chẳng hạn ).

Bước thứ sáu, cộng các giá trị ở bước năm lại với nhau. Rồi tiếp tục thực hiện với vector .

Mô hình tổng quát của các vector sau khi qua lớp Self-Attention



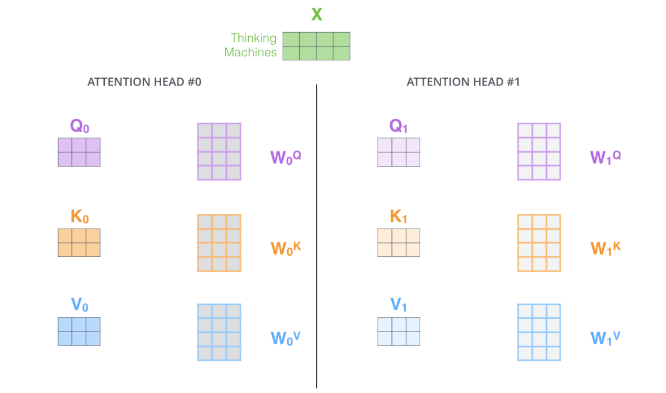
Hình Mô hình tổng quát các vector trong Self-Attention

( Nguồn <http://jalammar.github.io/illustrated-transformer/> )

#### **2.2.2.4 Mutil-head Attention**

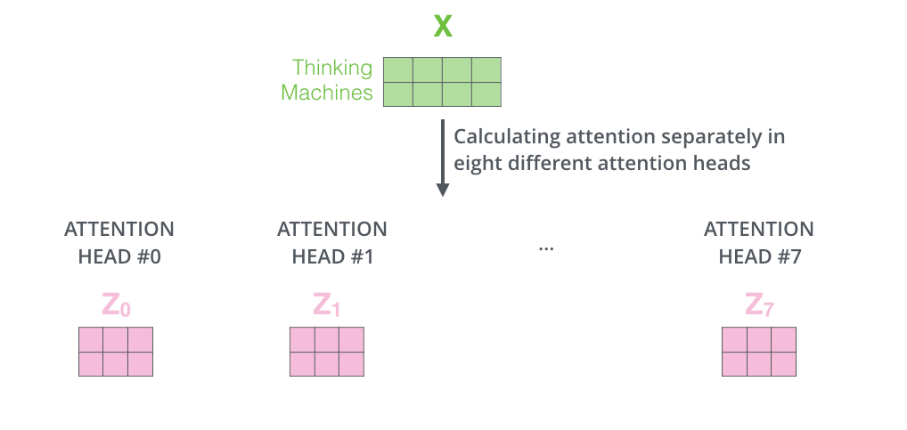
Ở cơ chế Self-Attention các từ chỉ tập chung vào chính nó,nhưng điều đó lại không mong muốn, cái cần là sự liên kết giữa những từ ở các vị trí khác nhau trong câu. Chính vì vậy Mutil-head Attention ra đời. Ý tưởng rất đơn giản là thay vì sử dụng 1 Attention (1 head) thì sử dụng nhiều Attention khác nhau ( Mutil-head) và biết đâu mỗi Attention chú ý đến một phần khác nhau trong câu.

Nó cung cấp cho lớp Attention nhiều “Không gian con biểu diễn”. Với Mutil-head Attention không chỉ có một mà có nhiều bộ ma trận trọng số Query/ Key/ Value ( Transformer sử dụng tám Attention head, vì vậy kết thức vố tám bộ mã hóa / giải mã).

Mỗi bộ này được khởi tạo ngẫu nhiên. Sau khi huấn luyện, mỗi tập hợp được sử dụng đối chiếu các Input Embedding ( hoặc vector từ bộ mã hóa/ giải mã thấp hơn ) và một không gian con khác. ( Hình 16)

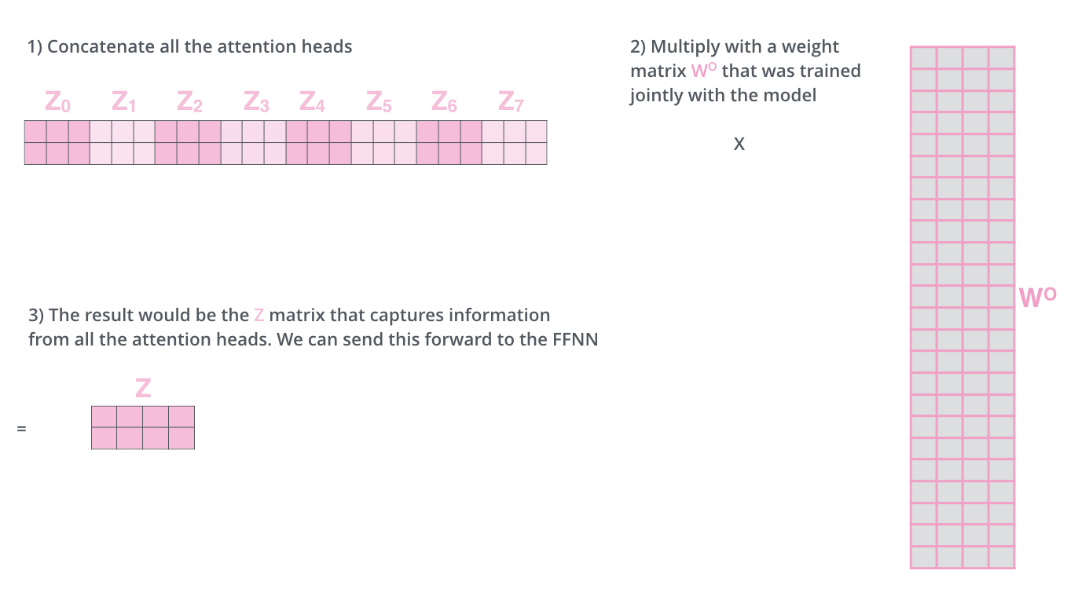
Hình 16 Bộ các vector Query/ Key/ Value

( Nguồn : <http://jalammar.github.io/illustrated-transformer/> )

Nếu khi thực hiện các tính toán Self-Attention tương tự đã nêu ở trên, thì sẽ có thể thu được 8 ma trận Z khác khau. ( Hình 17)

Hình 17 Các ma trận thu được

( Nguồn : <http://jalammar.github.io/illustrated-transformer/> )

Tuy nhiên lớp kế tiếp lại không mong nhận được 8 ma trận cùng 1 lúc mà chỉ cần 1 ma trận. Để giải quyết vấn đề này,có thể thực hiệ bằng cách nối các ma trận lại với nhau và sau đó nhân chúng với một ma trận trọng số bổ sưng (Hình 18).

Hình 18 Nối các ma trận tạo ma trận Z

( Nguồn : <http://jalammar.github.io/illustrated-transformer/> )

2.2.3 Decoder

* 1. Phương pháp đánh giá

Trình bày cách đánh giá cụ thể cho bài toán của các em

1. PHƯƠNG PHÁP THỰC HIỆN

Chương này trính bày các phương pháp để xây dựng hệ thống

**3.1 Các bước thực hiện**

|  |
| --- |
| Thu thập dữ liệu |
|  |
| Tiền xử lý dữ liệu |
|  |
| Huấn luyện mô hình |
|  |
| Sinh câu trả lời |
|  |
| Đánh giá và nhận xét |

Thu thập dữ liệu :

Với tập tiếng Anh sử dụng tập dữ liệu Cornell Movie-Dialogs Corpus kho dữ liệu này là các cuộc hội thoại bằng tiếng Anh được trích xuất từ các bộ phim thô.

Với tiếng Việt sử dụng lời sub với hơn 120.000 dòng từ Opensubtitles để xây dựn tập dữ liệu.

Tiền xử lý dữ liệu:

Làm sạch dữ liệu thu được bằng các loại bỏ các ký tự đặc biệt, xóa khoảng trắng và lưu lại file kết quả mới.

Huấn luyện mô hình :

Huấn luyện mô hình kết hợp sử dụng Transformer để đào tạo hệ thống.

Sinh câu trả lời :

Nhập dữ liệu đầu vào và ỉn ra kết quả.

Đánh giá và nhận xét :

Đánh giá kết quả thực hiện đã được được yêu cầu đề ra chưa, tiến hành nhận xét kết quả đó.

1. THỰC NGHIỆM

Chương này trình bày các bước để xây dựng mô hình

* 1. Xây dựng tập dữ liệu
  2. Tiền xử lý dữ liệu
  3. Huấn luyện mô hình
  4. Kiểm thử và điều chỉnh
  5. Đánh giá độ chính xác
  6. Thảo luận kết quả thực nghiệm

1. KẾT LUẬN

Trình bày kết quả của mô hình thực hiện được và hướng phát triển về sau.

* 1. Kết quả đạt được
  2. Hướng phát triển

TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | Hochreiter, Sepp, Jürgen Schmidhuber, "Long short-term memory," *Neural computation,* pp. 1735 - 1780, 1997. |
| [2] | Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin, "Attention Is All You Need," 2017. |

PHỤ LỤC

// Để các hình ảnh minh họa kết quả

1. <https://en.wikipedia.org/wiki/Artificial_neural_network> [↑](#footnote-ref-1)
2. <https://en.wikipedia.org/wiki/Feedforward_neural_network> [↑](#footnote-ref-2)
3. <https://en.wikipedia.org/wiki/Recurrent_neural_network> [↑](#footnote-ref-3)
4. <https://en.wikipedia.org/wiki/Long_short-term_memory> [↑](#footnote-ref-4)