

**ĐẠI HỌC QUỐC GIA HÀ NỘI**  
**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**

————— \* —————

**Phạm Minh Nguyên**

**NGHIÊN CỨU DỊCH MÁY TRUNG-VIỆT  
DỰA VÀO MÔ HÌNH TRANSFORMER**

**LUẬN VĂN THẠC SĨ HỆ THỐNG THÔNG TIN**

HÀ NỘI 08 – 2020

**ĐẠI HỌC QUỐC GIA HÀ NỘI**  
**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**

\_\_\_\_\_ \* \_\_\_\_\_

**Phạm Minh Nguyên**

**NGHIÊN CỨU DỊCH MÁY TRUNG-VIỆT  
DỰA VÀO MÔ HÌNH TRANSFORMER**

Ngành : Công nghệ thông tin

Chuyên ngành : Hệ thống thông tin

Mã số : 8480104.01

**LUẬN VĂN THẠC SĨ**

**NGƯỜI HƯỚNG DẪN KHOA HỌC: TS. Nguyễn Văn Vinh**

**HÀ NỘI 08 – 2020**

## **LỜI CAM ĐOAN**

Với mục đích học tập, nghiên cứu để nâng cao kiến thức và trình độ chuyên môn nên tôi đã làm luận văn này một cách nghiêm túc và hoàn toàn trung thực.

Trong luận văn tôi có sử dụng một số tài liệu tham khảo của một số tác giả. Tôi đã chú thích và nêu ra trong phần tài liệu tham khảo ở cuối luận văn.

Tôi xin cam đoan và chịu trách nhiệm về nội dung và sự trung thực trong luận văn tốt nghiệp Thạc sĩ của mình.

Hà Nội, ngày 20 tháng 08 năm 2020

Phạm Minh Nguyên

## LỜI CẢM ƠN

Lời đầu tiên, tôi xin chân thành cảm ơn các thầy cô giáo trong trường Đại Học Công Nghệ - Đại học Quốc Gia Hà Nội, đặc biệt là các thầy cô của khoa Công Nghệ Thông Tin đã truyền đạt cho tôi những kiến thức, kinh nghiệm vô cùng quý báu trong suốt thời gian qua.

Tôi xin gửi lời cảm ơn đến TS. Nguyễn Văn Vinh – giảng viên khoa Công Nghệ Thông tin – Trường Đại học Công Nghệ đã tận tình giúp đỡ, trực tiếp chỉ bảo và hướng dẫn tận tình trong suốt quá trình làm luận văn.

Cuối cùng, tôi xin được cảm ơn đến gia đình, bạn bè đã động viên, đóng góp ý kiến và giúp đỡ trong quá trình học tập, nghiên cứu và hoàn thành luận văn.

Do thời gian, kiến thức và kinh nghiệm của tôi còn hạn chế nên khóa luận không thể tránh khỏi những sai sót. Tôi hy vọng sẽ nhận được những ý kiến nhận xét, góp ý của các thầy cô giáo và các bạn để đề án được hoàn thiện hơn.

Tôi xin chân thành cảm ơn!

Hà Nội, ngày 20 tháng 08 năm 2020

Phạm Minh Nguyên

# MỤC LỤC

LỜI CAM ĐOAN .....	3
LỜI CẢM ƠN .....	4
MỤC LỤC .....	5
DANH MỤC CÁC KÝ HIỆU VÀ CHỮ VIẾT TẮT .....	7
DANH MỤC HÌNH VẼ .....	8
DANH MỤC BẢNG BIỂU .....	9
MỞ ĐẦU .....	10
CHƯƠNG 1: TỔNG QUAN VỀ DỊCH MÁY .....	12
1.1. Khái niệm dịch máy .....	12
1.2. Kiến trúc chung của một hệ dịch máy .....	13
1.3. Các cách tiếp cận dịch máy .....	14
1.3.1. Dịch máy thống kê .....	14
1.3.2. Dịch máy mạng nơ ron .....	15
1.4. Tiếng Trung Quốc và vấn đề dịch máy Trung – Việt .....	16
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT .....	18
2.1. Mạng nơ ron nhân tạo .....	18
2.1.1. Mạng nơ ron truyền thẳng .....	20
2.1.2. Mạng nơ ron hồi quy .....	21
2.1.3. Mạng bộ nhớ dài - ngắn (LSTM) .....	22
2.1.4. Huấn luyện mạng nơ ron .....	24
2.2. Word Embedding .....	25
2.2.1. Word2vec .....	26
2.2.2. GloVe .....	27
2.3. Mô hình seq2seq .....	28
2.4. Mô hình Transformer .....	31
2.4.1. Giới thiệu .....	31
2.4.2. Self-attention .....	31
2.4.3. Tổng quan mô hình .....	35

2.4.4. Bộ mã hóa.....	37
2.4.5. Bộ giải mã.....	40
2.4.6. Ứng dụng Attention trong mô hình Transformer .....	41
CHƯƠNG 3: DỊCH MÁY TRUNG-VIỆT DỰA VÀO MÔ HÌNH TRANSFORMER.....	42
3.1. Giới thiệu .....	42
3.2. Định hướng giải pháp .....	42
3.3. Thử nghiệm.....	42
3.3.1. Thử nghiệm mô hình Transformer .....	42
3.3.2. Thử nghiệm mô hình dịch máy nơ ron sử dụng RNN và Attention.....	44
3.4. Đánh giá.....	45
3.4.1. Phương pháp đánh giá .....	45
3.4.2. Kết quả.....	46
KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN .....	49
TÀI LIỆU THAM KHẢO .....	50

## DANH MỤC CÁC KÝ HIỆU VÀ CHỮ VIẾT TẮT

Ký hiệu viết tắt	Thuật ngữ đầy đủ	Giải thích
RNN	Recurrent neural network	Mạng nơ ron hồi quy
FFNN	Feed forward neural network	Mạng nơ ron truyền thẳng
NLP	Natural language processing	Xử lý ngôn ngữ tự nhiên
LSTM	Long short term memory	Mạng bộ nhớ dài ngắn
Word2Vec	Word to vector	
BOW	Bag of word	Mô hình túi từ
CBOW	Continue bag of word	Mô hình túi từ liên tiếp
GloVe	Global vector	Mô hình vector toàn cục
BLEU	Bilingual Evaluation Understudy Score	Điểm đánh giá chất lượng dịch máy

# DANH MỤC HÌNH VẼ

Hình 1.1: Quá trình xử lý tài liệu dịch máy.....	12
Hình 1.2: Các loại hệ thống dịch máy. ....	13
Hình 1.3: Kiến trúc mã hóa – giải mã. ....	15
Hình 1.4: Ví dụ sắp xếp từ trong tiếng Trung và tiếng Việt.....	16
Hình 2.1: Mô hình mạng nơ ron đơn giản.....	18
Hình 2.2: Ví dụ về một nơ ron nhân tạo.....	19
Hình 2.3: Một số hàm kích hoạt thông dụng.....	20
Hình 2.4: Mạng nơ ron truyền thẳng.....	21
Hình 2.5: Mạng nơ ron hồi quy.....	22
Hình 2.6: Mạng bộ nhớ dài ngắn.....	23
Hình 2.7: Cổng quên.....	23
Hình 2.8: Cổng vào.....	24
Hình 2.9: Cổng ra.....	24
Hình 2.10: Cấu trúc huấn luyện mạng nơ ron.....	25
Hình 2.11: Biểu diễn Word Embedding.....	26
Hình 2.12: Mô hình CBOW và Skip-grams.....	27
Hình 2.13: Kiến trúc của mô hình Seq2Seq.....	28
Hình 2.14: Minh họa mô hình seq2seq dung trong bài toán dịch máy.....	30
Hình 2.15: Quá trình tính toán vector attention.....	32
Hình 2.16: Kiến trúc mô hình Transformer.....	36
Hình 2.17: Bộ mã hóa và giải mã trong mô hình transformer.....	37
Hình 2.18: Một lớp trong bộ mã hóa của mô hình Transformer.....	38
Hình 2.19: Ví dụ biểu diễn từ đầu vào.....	39
Hình 2.20: Quá trình tính toán vector attention với nhiều “head”.....	39
Hình 2.21: Bộ giải mã của mô hình transformer.....	40



## DANH MỤC BẢNG BIỂU

Bảng 3-1: Thông tin cấu hình phần cứng .....	42
Bảng 3-2: Thông kê về dữ liệu sử dụng .....	43
Bảng 3-3: Các tham số huấn luyện mô hình Transformer .....	44
Bảng 3-4: Các tham số huấn luyện sử dụng RNN và Attention.....	44
Bảng 3-5: Điểm BLEU của hệ thống dịch máy Trung – Việt .....	46
Bảng 3-6: Một số kết quả dịch .....	46

# MỞ ĐẦU

Cùng với sự phát triển của trong quan hệ kinh tế Trung – Việt, số lượng các văn bản dịch Trung – Việt ngày càng lớn, do đó nhu cầu đặt ra là cần thiết kể một mô hình tự động để hỗ trợ dịch thuật. Trước đây dịch máy được thực hiện theo từ hay cụm từ, tức là dựa vào hàng triệu từ hay cụm từ đã được dịch để đối chiếu, so sánh và chọn cụm từ nào sát nhất bằng phương pháp thống kê để đưa vào kết quả. Hiện nay dịch máy có thể thực hiện theo cả câu, rồi dùng ngữ cảnh để quyết định xem từ đó trong ngữ cảnh đó thì chọn nghĩa nào cho chính xác nhất. Các nghiên cứu do đó cũng chuyển dần sang dịch máy nơ ron (Neural Machine Translation), đây là cách tiếp cận dịch máy phổ biến trong những năm gần đây và đã cho ra các kết quả thực sự tốt, tới mức ngang hoặc hơn cả con người.

RNN, LSTM, GRU là các phương pháp tiếp cận hiện đại trong mô hình ngôn ngữ và dịch máy, từ đó khắc phục được những hạn chế của việc phụ thuộc xa trong mạng nơ ron truyền thống. Tuy nhiên trong nhiều bài toán về dịch thuật, việc cải thiện cũng không đáng kể. Chính vì thế kỹ thuật attention được áp dụng để mang lại hiệu quả cao hơn. Cách tiếp cận sequence-to-sequence with attention là một trong những mô hình đầu tiên áp dụng kỹ thuật attention kết hợp với LSTM. Năm 2017, các kỹ sư của Google đã giới thiệu kỹ thuật self-attention và đề xuất mô hình Transformer, cho phép thay thế hoàn toàn kiến trúc recurrent của mô hình RNN bằng các mô hình full-connected. Dịch máy lúc này chỉ hoàn toàn dựa vào kỹ thuật attention.

Tại Việt Nam, vấn đề dịch máy cũng đang rất được quan tâm. Tuy nhiên, các nghiên cứu về tiếng Việt còn khá ít. Khoảng những năm trở lại đây có một số nhóm nghiên cứu về dịch máy tiếng Việt nhưng chủ yếu tập trung vào hệ dịch Anh-Việt, Pháp-Việt. Hiện nay Google là hệ thống dịch mở được sử dụng nhiều nhất trên thế giới đã tích hợp tiếng Việt vào hệ thống của họ. Hệ dịch mở Google dịch khá tốt giữa tiếng Anh với các ngôn ngữ khác, tuy nhiên với các cặp ngôn ngữ khác như Trung-Việt Google sử dụng tiếng Anh làm trung gian nên chất lượng dịch còn khá thấp

Trong phạm vi khóa luận sẽ trình bày về mô hình Transformer – một mô hình dịch máy hình hoàn toàn chỉ dựa vào kỹ thuật attention và ứng dụng vào dịch máy Trung-Việt.

Luận văn có bố cục gồm 3 chương chính:

## **Chương 1:** Tổng quan về dịch máy

Chương này giới thiệu tổng quan về dịch máy, một số cách tiếp cận dịch máy, tiếng Trung và vấn đề dịch máy Trung – Việt hiện nay

## **Chương 2:** Cơ sở lý thuyết

Chương này đi sâu tìm hiểu về mô hình mạng nơ ron nhân tạo và mô hình Transformer sẽ áp dụng trong khóa luận

### **Chương 3:** Dịch máy Trung-Việt dựa vào mô hình Transformer

Chương này sẽ trình bày việc áp dụng mô hình Transformer trong dịch máy Trung – Việt và các kết quả thực nghiệm

Cuối cùng là một số kết luận và hướng phát triển trong tương lai

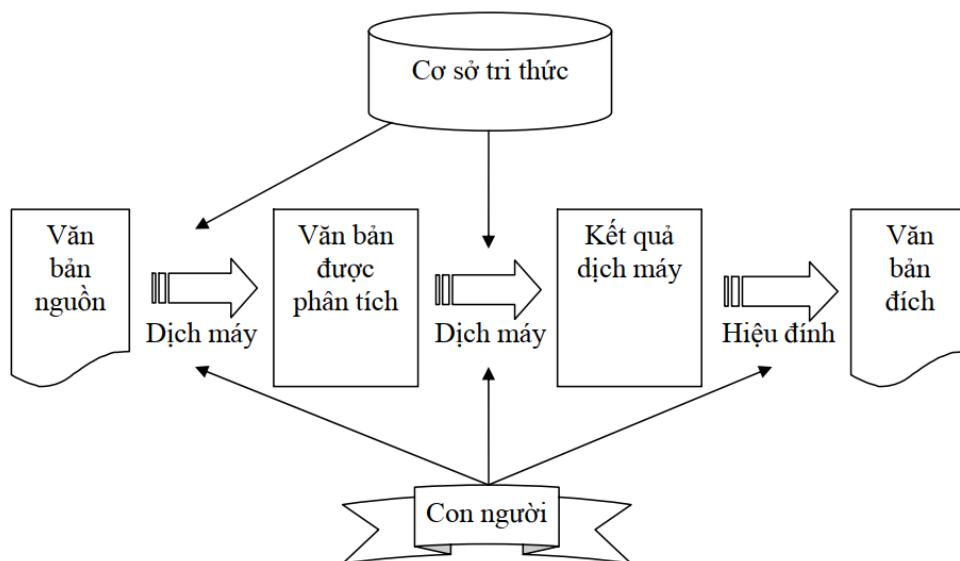
# CHƯƠNG 1: TỔNG QUAN VỀ DỊCH MÁY

## 1.1. Khái niệm dịch máy

Dịch máy (machine translation) được hiểu là việc thực hiện dịch một ngôn ngữ này (gọi là ngôn ngữ nguồn) sang một hoặc nhiều ngôn ngữ khác (gọi là ngôn ngữ đích) một cách tự động, không có sự can thiệp của con người trong quá trình dịch [10].

Hiện nay, trên thế giới có khoảng 5650 ngôn ngữ khác nhau, với số lượng ngôn ngữ lớn như vậy đã gây ra rất nhiều khó khăn trong việc trao đổi thông tin. Để có thể trao đổi thông tin phải cần đến một đội ngũ phiên dịch khổng lồ để dịch các văn bản, tài liệu, lời nói từ tiếng này sang tiếng khác. Vì vậy, con người đã nghĩ đến việc thiết kế một hệ thống tự động trong việc dịch.

Ngày nay, có khoảng 5000 đến 7000 ngôn ngữ khác nhau trên thế giới. Với số lượng như vậy đã gây ra rất nhiều vấn đề trong giao tiếp và cản trở sự phát triển của thương mại quốc tế. Cùng với đó là sự bùng nổ thông tin cũng là gia tăng theo cấp số nhân lượng dữ liệu cần dịch. Để đáp ứng được nhu cầu này, cần phải có nhiều hơn nữa các dịch giả. Với nhu cầu như vậy, con người đã nghĩ đến việc thiết kế một mô hình tự động để hỗ trợ dịch thuật. Khái niệm dịch máy xuất hiện kể từ khi máy tính ra đời và phát triển cho đến ngày nay. Các nghiên cứu trong lĩnh vực dịch máy và các công nghệ phần cứng liên tục phát triển để giải quyết ngày càng nhiều lĩnh vực trong thực tế. Mặc dù thực tế, việc thành công bị giới hạn trong một số lĩnh vực cụ thể, ví dụ như lĩnh vực dự báo thời tiết từ ngữ rõ ràng, dễ hiểu hơn ngành y học. Việc dịch máy được khuyến khích sử dụng vì chúng có thể tự học và việc đưa vào cho nhiều người sử dụng cũng là một phần rất quan trọng trong sự phát triển của hệ thống dịch máy.

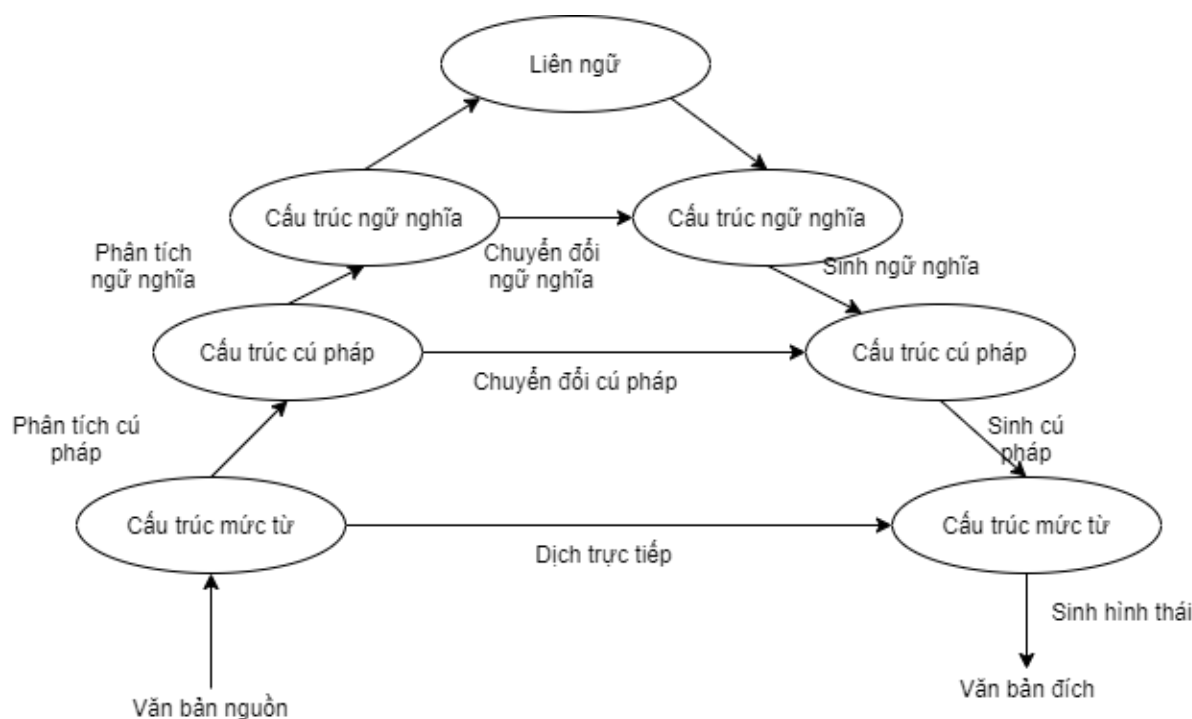


**Hình 1.1: Quá trình xử lý tài liệu dịch máy.**

Đầu vào của một hệ dịch máy là một văn bản được viết bằng một ngôn ngữ nguồn, quá trình dịch có thể chia thành hai giai đoạn. Đầu tiên, văn bản được phân tích thành các thành phần, sau đó được dịch thành văn bản ở dạng ngôn ngữ đích. Kết quả dịch có thể được con người hiệu chỉnh để trở thành bản dịch tốt. Như vậy trong một quá trình dịch, con người có thể tác động vào các bước xử lý với mục đích làm cho kết quả dịch tốt hơn.

## 1.2. Kiến trúc chung của một hệ dịch máy

Kiến trúc hiện thời của một hệ dịch máy có thể được phân thành 3 lớp chính sau: trực tiếp, chuyển đổi, và liên ngữ. Ba lớp này tương ứng với các loại khác nhau phụ thuộc vào mức độ phân tích của hệ thống.



**Hình 1.2: Các loại hệ thống dịch máy.**

**Kiến trúc dịch trực tiếp (Direct Architecture):** Thay thế từng từ trong văn bản nguồn thành từ trong văn bản đích rồi sinh trở lại văn bản đích theo đúng thứ tự văn bản nguồn. Kiến trúc này đơn giản nhưng không đạt hiệu quả cao vì các khác biệt về cú pháp và tính đa nghĩa của từ. Kiến trúc này được áp dụng vào những năm đầu của lịch sử dịch máy và đạt được thành công trong những ngữ cảnh hẹp hoặc trong bài toán đòi hỏi chất lượng không cao lắm.

**Kiến trúc dịch chuyển đổi (Transfer Architecture):** Gồm hai mức, chuyển đổi cú pháp và chuyển đổi ngữ nghĩa, thực hiện chuyển đổi các tri thức ngôn ngữ từ ngôn ngữ nguồn sang ngôn ngữ đích (từ, cú pháp, nghĩa, ...). Kiến trúc này có độ chính xác cũng như dễ đọc dễ hiểu, giải quyết mập mờ tốt hơn nhiều kiến trúc dịch trực tiếp.

Tuy vậy thường phải tốn nhiều công sức trong việc tiến hành việc chuyển đổi tri thức ngôn ngữ cho từng cặp ngôn ngữ.

**Kiến trúc dịch qua ngôn ngữ trung gian (Interlingual Architecture):** Phân tích ngôn ngữ nguồn và đưa ra mô tả về tri thức ngôn ngữ tương ứng trên một ngôn ngữ trung gian, độc lập với ngôn ngữ nguồn. Từ đó tạo ra văn bản cho ngôn ngữ đích. Nói cách khác, các cặp ngôn ngữ nguồn và đích đều được dịch thông qua một ngôn ngữ trung gian. Đây là kiến trúc hoàn hảo mà các hệ dịch máy vươn tới.

### 1.3. Các cách tiếp cận dịch máy

Có nhiều các tiếp cận dịch máy: dựa trên luật (rule-based), dựa trên cụm từ (phrase-based), dựa trên thống kê (statistics-based), dựa trên mạng nơ ron , .v.v. Các triển khai hệ thống dịch trong thực tế không phải luôn luôn sử dụng chỉ một hướng tiếp cận, nhiều hệ thống kết hợp các phương pháp tiếp cận khác nhau để đạt được kết quả tốt nhất. Dưới đây là một số các cách tiếp cận dịch máy:

#### 1.3.1. Dịch máy thống kê

Dịch máy thống kê [3] là một cách tiếp cận dịch thuật bằng cách xây dựng các mô hình xác suất, sau đó kết hợp các mô hình này để chọn ra bản dịch khả thi nhất. Ý tưởng của mô hình này là thay vì xây dựng từ điển, quy luật bằng tay, hệ dịch sẽ tự động xây dựng từ điển và quy luật dựa trên thống kê. Cách tiếp cận này không đòi hỏi sự phân tích sâu về ngôn ngữ, chúng thực hiện hoàn toàn tự động các quá trình phân tích, chuyển đổi, tạo câu dựa trên kết quả thống kê có được từ kho ngữ liệu.

Cách tiếp cận dịch máy dựa trên thống kê được Brown và các cộng sự đưa ra từ những năm đầu thập kỷ 1990 sau khi thấy được những thành công của việc áp dụng thống kê trong một vài lĩnh vực. Brown và các cộng sự giả định rằng mỗi câu ở một ngôn ngữ sẽ có được những câu dịch khác nhau ở ngôn ngữ khác. Và họ đã đưa ra xác suất  $P(V|C)$  là xác suất điều kiện để dịch được câu  $V$  ở ngôn ngữ đích khi đã có câu  $C$  ở ngôn ngữ nguồn.

Ý tưởng cơ bản của cách tiếp cận này là từ một câu  $C$  ở ngôn ngữ nguồn, hệ thống đi tìm một câu  $V$  ở ngôn ngữ đích sao cho xác suất  $P(V|C)$  đạt giá trị lớn nhất.

Ví dụ chúng ta muốn dịch câu tiếng Trung  $C = c_1, c_2, \dots, c_n$  sang tiếng Việt. Trong các câu tiếng Việt, ta chọn câu  $\tilde{V} = v_1, v_2, \dots, v_n$  có xác suất  $P(V|C)$  lớn nhất

$$\begin{aligned}\tilde{V} &= \operatorname{argmax}_v P(V|C) = \operatorname{argmax}_v \frac{P(C|V)P(V)}{P(C)} \\ &\sim \operatorname{argmax}_v P(C|V)P(V)\end{aligned}\tag{1}$$

Bởi ngôn ngữ nguồn không thay đổi, do vậy  $P(C)$  là hằng số, do vậy chúng ta có thể bỏ qua phần mẫu số ở (1). Ở đây chúng ta cần quan tâm đến ba vấn đề sau:

- Mô hình ngôn ngữ  $P(V)$
- Mô hình dịch  $P(C|V)$
- Bộ giải mã (decoder) để tìm kiếm một câu dịch có giá trị  $P(V|C)$  lớn nhất

Hiệu suất của các hệ thống dịch máy thông kê phụ thuộc nhiều vào cặp ngôn ngữ đưa ra, hiệu suất của hệ thống này đạt được tốt nhất trên những cặp ngôn ngữ có cấu trúc ngữ pháp tương đồng, chẳng hạn như tiếng Anh và tiếng Pháp. Các cặp ngôn ngữ có sự khác biệt lớn về cấu trúc ngữ pháp, đặc biệt các ngôn ngữ không có nguồn gốc châu Âu như tiếng Trung Quốc, tiếng Nhật... thì hiệu suất giảm đáng kể.

Các nghiên cứu về dịch máy thông kê gần đây cho thấy các mô hình dịch máy nơ ron đạt được các kết quả tốt hơn so với mô hình dịch máy thông kê truyền thống. Phần dưới đây sẽ trình bày về cách tiếp cận dịch máy mạng nơ ron

### 1.3.2. Dịch máy mạng nơ ron

Đây là một cách dịch tương tự như cách dịch của con người. Tức là hệ thống sẽ thực hiện đọc trọn vẹn một câu nguồn, “hiểu” ý nghĩa của nó, sau đó sẽ tiến hành dịch sang ngôn ngữ đích.



**Hình 1.3: Kiến trúc mã hóa – giải mã.**

Đầu tiên hệ dịch nơ ron sử dụng bộ mã hóa (Encoder) để đọc toàn bộ câu nguồn và mã hóa nó dưới dạng một vector biểu diễn ý nghĩa. Sau đó, bộ giải mã (Decoder) sẽ đọc và giải mã vector biểu diễn câu nguồn này để sinh ra bản dịch tương ứng sang ngôn ngữ đích, quá trình mã hóa - giải mã được minh họa như ở hình 1.3.

Tương tự như dịch máy thông kê, dịch máy dựa trên mạng nơ ron là mô hình dịch máy dựa trên dữ liệu, phụ thuộc vào dữ liệu song ngữ sử dụng để huấn luyện

Trong luận văn sẽ sử dụng hướng tiếp cận dịch máy nơ ron và áp dụng vào cặp ngôn ngữ Trung – Việt. Chi tiết về mô hình mã hóa và giải mã được sử dụng sẽ được trình bày trong chương 2

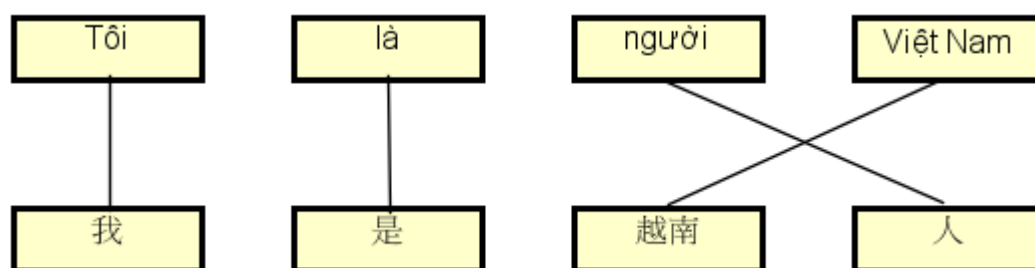
## 1.4. Tiếng Trung Quốc và vấn đề dịch máy Trung – Việt

Dịch thuật là một vấn đề khó, bởi nó đòi hỏi cần phải có kiến thức về cả ngôn ngữ nguồn và ngôn ngữ đích. Bên cạnh đó, mỗi một ngôn ngữ lại có những đặc điểm riêng khiến cho việc chuyển nghĩa hoàn toàn các từ là vấn đề khó.

Tiếng Việt và tiếng Trung đều thuộc ngôn ngữ đơn lập, dấu cách (space) không được sử dụng như một kí hiệu phân tách từ, nó chỉ có ý nghĩa phân tách các âm tiết với nhau, một từ có thể bao gồm một hoặc nhiều từ chính tả.

Một câu tiếng Trung bao gồm một dãy các từ chính tả, kể cả dấu câu, nằm liên tiếp với nhau và không có khoảng trắng giữa các từ chính tả này [9]. Trong tiếng Việt, các từ chính tả được phân cách với nhau bởi một khoảng trắng, các dấu câu nằm liền sau từ chính tả [1]. Ví dụ, chúng ta có cặp câu như hình 1.2:

- Câu “我是越南人” có thể tách thành các từ [我, 是, 越南, 人] và giữa các từ không hề có dấu cách
- Câu tiếng Trung trên được dịch ra là “Tôi là người Việt Nam”, bao gồm các từ [Tôi, là, người, Việt Nam]



**Hình 1.4: Ví dụ sắp xếp từ trong tiếng Trung và tiếng Việt.**

Bên cạnh việc phân tách từ, ngữ pháp cũng là một vấn đề quan trọng. Có những ngôn ngữ có hệ thống ngữ pháp khá chặt chẽ, trong khi tiếng Việt có cấu trúc lỏng lẻo hơn. Thậm chí còn có sự khác biệt trong vị trí của chủ ngữ (S), động từ (V), bổ ngữ (O), ví dụ như tiếng Anh, tiếng Pháp, tiếng Trung Quốc, tiếng Việt là những ngôn ngữ mà động từ đứng ở giữa chủ ngữ và bổ ngữ, trong khi tiếng Ireland và tiếng Ả Rập động từ nằm ở trước chủ ngữ.

Ngoài ra sự đa nghĩa là điều thường xuyên có trong một ngôn ngữ, ví dụ như 旅行 (luxing): du lịch 银行 (yinghang): ngân hàng, hai từ này cách viết hoàn toàn giống nhau, nhưng có thể phát âm theo hai cách khác nhau, dẫn đến nghĩa khác nhau. Do vậy nghĩa của một từ sẽ mơ hồ nếu bối cảnh không rõ ràng. Đối với một số trường hợp, một người có thể sử dụng những từ “đa dụng” như “ấy”, “nó” ... khiến việc tìm kiếm nghĩa của từ trở nên khó khăn. Một vấn đề nữa, một ngôn ngữ có thể



chứa các từ mà ta không thể tìm thấy nghĩa chính xác trong ngôn ngữ khác, ví dụ như một số từ lóng như “thần sầu”, “vi diệu”, ...

Mối quan hệ giữa Trung Quốc – Việt Nam là mối quan hệ lâu đời và đã có lịch sử hàng nghìn năm. Bên cạnh việc là ngôn ngữ có cấu trúc chủ ngữ, động từ, bổ ngữ, cả hai ngôn ngữ đều có chung một lượng lớn các từ “Hán – Việt”. Mặc dù không phải tất cả các ký tự tiếng Trung Quốc đều có thể chuyển đổi sang tiếng Việt, nhưng nó chiếm một phần quan trọng trong tiếng Việt khi mà một phần ba số từ vựng tiếng Việt là tiếng Hán – Việt. Ví dụ, chỉ cần chuyển đổi một thành ngữ tiếng Trung Quốc “半信半疑”, chúng ta “bán tín bán nghi” – hoàn toàn là thành ngữ Việt Nam. Các tên người tiếng Trung hoàn toàn có thể dịch trực tiếp sang tiếng Việt và cũng có rất nhiều văn bản, thơ cổ ... tồn tại ở cả ba dạng là Hán tự, Hán – Việt, tiếng Việt.

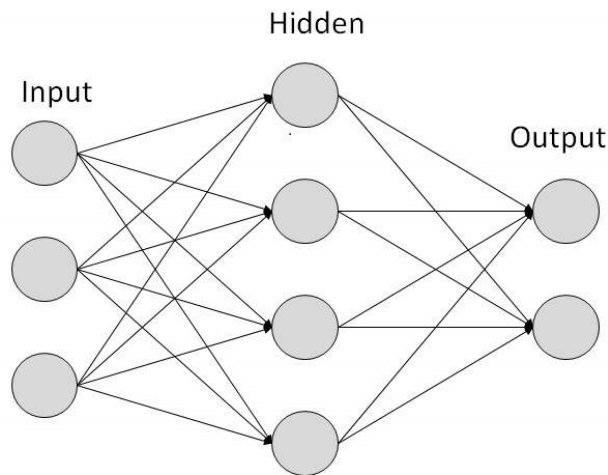
Bên cạnh những điểm chung như cùng là ngôn ngữ đơn lập, cấu trúc ngữ pháp lỏng lẻo, từ Hán – Việt, thì sự khác biệt giữa hai ngôn ngữ vẫn còn khá lớn so với sự giống nhau như là ký tự tượng hình, chữ viết hoa, viết thường ... và nó gây ra nhiều vấn đề trong dịch thuật. Ngoài ra ngày nay các nghiên cứu về dịch máy Trung – Việt vẫn đang còn hạn chế và ít tài liệu tham khảo.

Luận văn sẽ nghiên cứu để xây dựng một hệ thống dịch máy sử dụng mô hình Transformer ứng dụng cho cặp ngôn ngữ Trung – Việt.

## CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

### 2.1. Mạng nơ ron nhân tạo

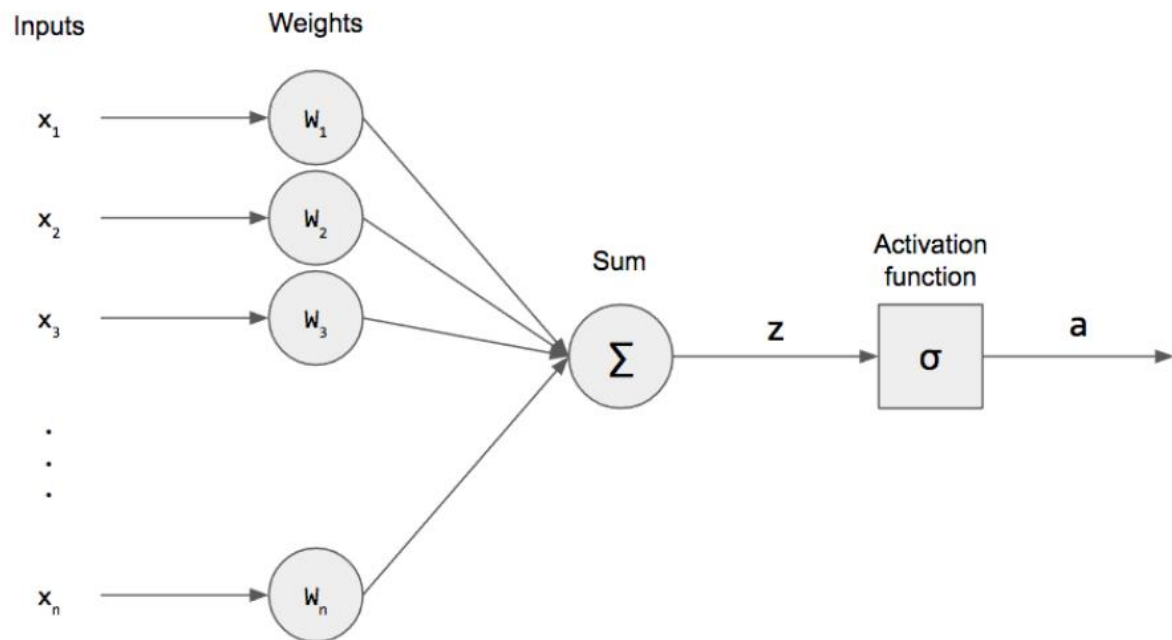
Mạng nơron nhân tạo [5] là một mô hình xử lý thông tin phỏng theo cách thức xử lý thông tin của các hệ nơron sinh học. Hình 2.3 cho thấy một mạng nơ-ron đơn giản được tạo nên bởi tầng vào, tầng ra và tầng ẩn. Từ “ẩn” có nghĩa là chúng ta có thể quan sát đầu vào và đầu ra trong khi cấu trúc kết nối chúng vẫn bị ẩn.



**Hình 2.1: Mô hình mạng nơ ron đơn giản**

Mạng nơron nhân tạo được tạo nên từ một số lượng lớn các phần tử (nơron) kết nối với nhau thông qua các liên kết (trọng số liên kết) làm việc như một thể thống nhất để giải quyết một vấn đề cụ thể nào đó. Một mạng nơron nhân tạo được cấu hình cho một ứng dụng cụ thể (nhận dạng mẫu, phân loại dữ liệu,...) thông qua một quá trình học từ tập các mẫu huấn luyện. Về bản chất “học” chính là quá trình hiệu chỉnh trọng số liên kết giữa các nơron. Dưới đây sẽ trình bày chi tiết về một nơ ron đơn lẻ và cụ thể về mô hình mạng nơ ron.

Lấy ý tưởng từ nơ ron sinh học của não người, một nơ ron có thể nhận nhiều đầu vào và cho ra một kết quả duy nhất.



**Hình 2.2: Ví dụ về một nơ ron nhân tạo.**

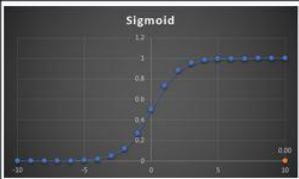
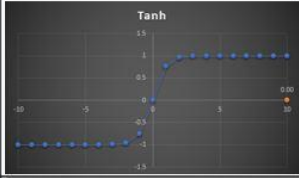
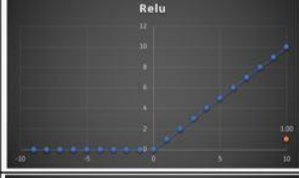

Mỗi nơ ron sẽ nhận một hoặc nhiều đầu vào  $x$  dạng nhị phân và cho ra một kết quả  $a$  dạng nhị phân duy nhất. Các đầu vào sẽ được điều chỉnh sự ảnh hưởng nhờ trọng số  $w$  của nó, còn kết quả đầu ra sẽ được quyết định dựa trên một ngưỡng  $b$  nào đó

$$a = \begin{cases} 0 & \text{Nếu } \sum_i w_i x_i \leq \text{threshold} \\ 1 & \text{Nếu } \sum_i w_i x_i > \text{threshold} \end{cases}$$

Đặt  $b = -\text{threshold}$  biểu thức trên có thể viết lại thành:

$$a = \begin{cases} 0 & \text{Nếu } \sum_i w_i x_i + b \leq 0 \\ 1 & \text{Nếu } \sum_i w_i x_i + b > 0 \end{cases}$$

Với đầu vào và đầu ra dạng nhị phân, rất khó để điều chỉnh một lượng nhỏ đầu vào để đầu ra thay đổi chút ít, do vậy để linh động, chúng ta có thể mở rộng chúng ra cả khoảng  $[0, 1]$ . Lúc này chúng ta sử dụng một hàm đặc biệt, gọi là hàm kích hoạt để giới hạn giá trị đầu ra. Các hàm kích hoạt thường rất đa dạng, có thể là hàm tuyến tính hoặc phi tuyến. Việc lựa chọn hàm kích hoạt nào là tùy thuộc vào từng bài toán và kinh nghiệm của người thiết kế mạng. Một số hàm kích hoạt thường sử dụng trong các mô hình mạng nơron được đưa ra trong hình dưới.

Name	Plot	Equation	Derivative
Sigmoid		$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
Tanh		$f(x) = \tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$	$f'(x) = 1 - f(x)^2$
Rectified Linear Unit (relu)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Leaky Rectified Linear Unit (Leaky relu)		$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0.01 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$

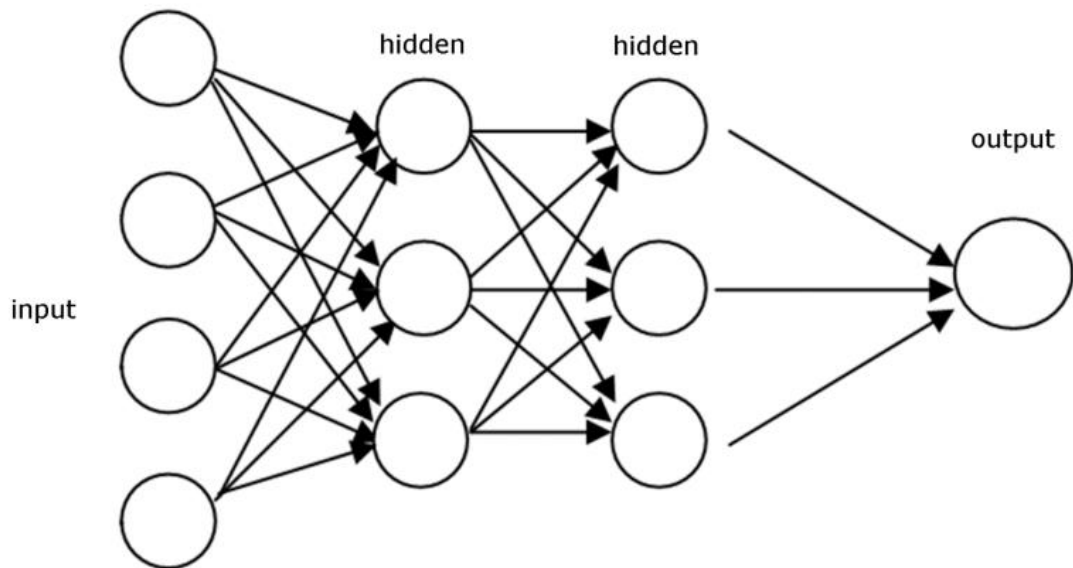
**Hình 2.3: Một số hàm kích hoạt thông dụng.**

Như vậy nơron nhân tạo nhận các tín hiệu đầu vào, xử lý (nhân các tín hiệu này với trọng số liên kết, tính tổng các tích thu được rồi gửi kết quả tới hàm kích hoạt), và cho một tín hiệu đầu ra (là kết quả của hàm kích hoạt).

Một mạng nơron là sự kết hợp của các tầng nơron như hình 2.3. Dựa trên cách thức liên kết các nơron người ta chia làm hai loại: Mạng nơron truyền thẳng và mạng nơron hồi quy.

### **2.1.1. Mạng nơron truyền thẳng**

Mạng nơron truyền thẳng là một mạng nơron nhân tạo, trong đó các kết nối giữa các nút không tạo thành chu kỳ. Mô hình mạng nơron chuyển tiếp là hình thức đơn giản nhất của mạng nơron vì thông tin chỉ được xử lý theo một hướng. Mặc dù dữ liệu có thể đi qua nhiều nút ẩn, nó luôn đi theo một hướng và không bao giờ lùi. Số đặc trưng của tập dữ liệu sẽ tương ứng với số nơron trong lớp đầu vào. Tất cả các nơron này được kết nối với mỗi nơron trong lớp ẩn thông qua các đường liên kết gọi là “khớp thần kinh”. Mỗi “khớp thần kinh” sẽ được gán một trọng số (weight). Các trọng số này sẽ được điều chỉnh trong quá trình học của mạng nơron nhân tạo để mô hình hóa mối liên hệ giữa lớp đầu vào và đầu ra.



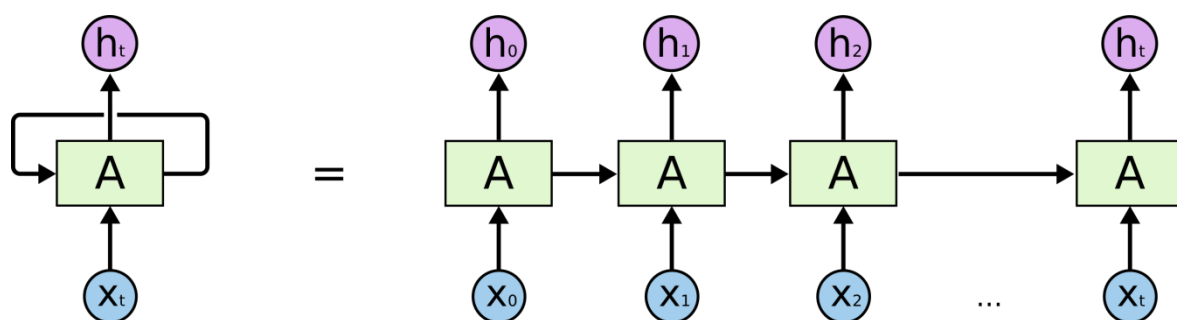
**Hình 2.4: Mạng nơ ron truyền thẳng.**

Trái ngược của một mạng nơ ron truyền thẳng là một mạng nơ ron hồi quy (RNN)

### **2.1.2. Mạng nơ ron hồi quy**

Như đã đề cập ở trên, mạng nơron gồm 3 tầng chính là tầng vào, tầng ra và tầng ẩn. Có thể thấy đầu vào và đầu ra của mạng neuron này là độc lập với nhau. Như vậy mô hình này không phù hợp với những bài toán dạng chuỗi như mô tả, hoàn thành câu, ... vì những dự đoán tiếp theo như từ tiếp theo phụ thuộc vào vị trí của nó trong câu và những từ đằng trước nó. Và như vậy, RNN ra đời với ý tưởng chính là sử dụng một bộ nhớ để lưu lại thông tin từ từ những bước tính toán xử lý trước để dựa vào nó có thể đưa ra dự đoán chính xác nhất cho bước dự đoán hiện tại.

Xét một nơron hồi quy A. Nó nhận đầu vào là  $x_t$ , tiến hành xử lý và đưa ra đầu ra là  $h_t$ . Điểm đặc biệt của A là nó sẽ lưu lại giá trị  $h_t$  để sử dụng cho đầu vào tiếp theo. Đầu ra  $y_t$  được tính dựa trên giá trị  $h_t$  sử dụng hàm kích hoạt softmax.



**Hình 2.5: Mạng nơ ron hồi quy.**

RNN có thể được mô tả bởi hai hàm sau:

$$h_t = \sigma(W^{hx}x_t + W^{hh}h_{t-1} + b_h)$$

$$y_t = \text{softmax}(W^{yh}h_t + b_y)$$

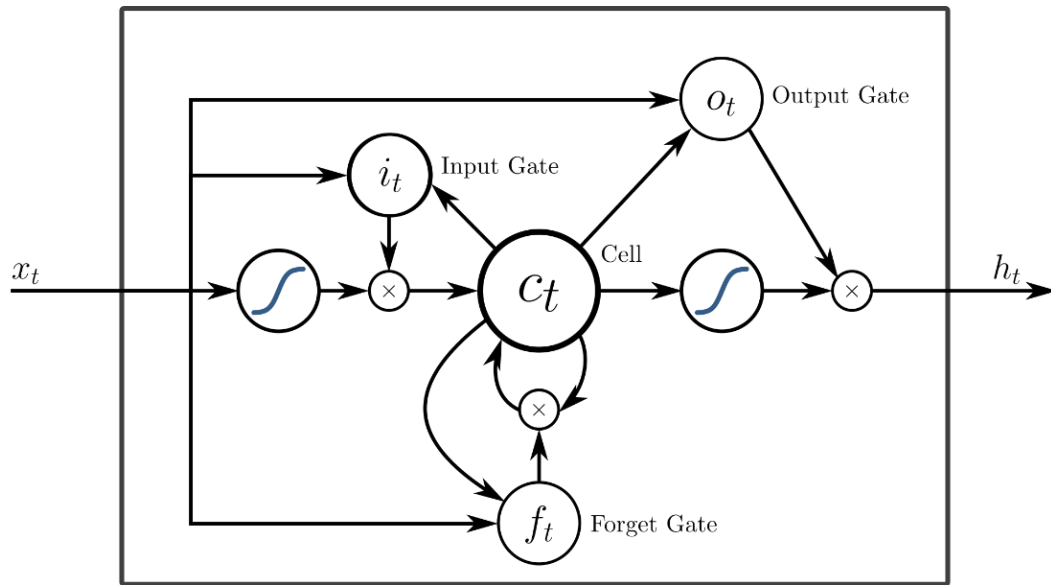
Trong đó  $W^{hx}$  là ma trận trọng số giữa lớp đầu vào và lớp ẩn,  $W^{hh}$  là ma trận trọng số lặp lại giữa lớp ẩn và chính nó,  $W^{yh}$  là ma trận trọng số giữa lớp ẩn và lớp đầu ra,  $b_h$  và  $b_y$  là các tham số điều chỉnh.

Một điểm nổi bật của RNN chính là ý tưởng kết nối các thông tin phía trước để dự đoán cho hiện tại. Đôi lúc ta chỉ cần xem lại thông tin vừa có thôi là đủ để biết được tình huống hiện tại. Ví dụ, ta có câu: “*các đám mây trên bầu trời*” thì ta chỉ cần đọc tới “*các đám mây trên bầu*” là đủ biết được chữ tiếp theo là “*trời*” rồi. Trong tình huống này, khoảng cách tới thông tin có được cần để dự đoán là nhỏ, nên RNN hoàn toàn có thể học được. Nhưng trong nhiều tình huống ta buộc phải sử dụng nhiều ngữ cảnh hơn để suy luận. Ví dụ, dự đoán chữ cuối cùng trong đoạn: “*I grew up in France... I speak fluent French.*”. Rõ ràng là các thông tin gần (“*I speak fluent*”) chỉ có phép ta biết được đằng sau nó sẽ là tên của một ngôn ngữ nào đó, còn không thể nào biết được đó là tiếng gì. Muốn biết là tiếng gì, thì ta cần phải có thêm ngữ cảnh “*I grew up in France*” nữa mới có thể suy luận được. Rõ ràng là khoảng cách thông tin lúc này có thể đã khá xa rồi.

Theo [Hochreiter \(1991\)](#) và [Bengio \(1994\)](#), với khoảng cách càng lớn dần thì RNN bắt đầu không thể nhớ và học được nữa. Và LSTM ra đời dựa trên RNN, có khả năng giải quyết vấn đề này.

### 2.1.3. Mạng bộ nhớ dài - ngắn (LSTM)

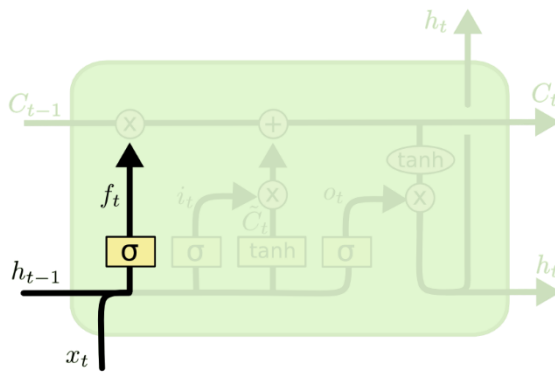
Long Short Term Memory networks – thường được gọi là “LSTM”, là trường hợp đặc biệt của RNN, có khả năng học với sự phụ thuộc lâu dài của các nơ-ron. Mô hình này được giới thiệu bởi Hochreiter & Schmidhuber (1997), và được cải tiến lại bởi Ayako Mikami (2016). Mục tiêu chính của LSTM là quyết định thông tin nào được lưu lại và loại bỏ tại mỗi nơ-ron của RNN.



**Hình 2.6: Mạng bộ nhớ dài ngắn.**

Chìa khóa của LSTM là trạng thái tế bào (cell state). 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) như hình 2.5:

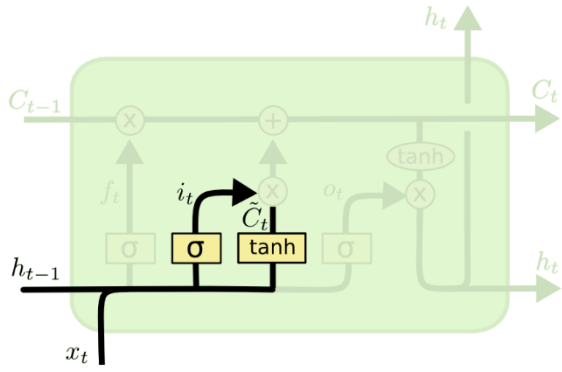
**Cổng quên:** Cổng này quyết định xem thông tin nào trong bộ nhớ hiện tại được giữ và thông tin nào bị bỏ lại. Thông tin đầu vào được cho vào hàm sigmoid. Đầu ra của hàm này đóng vai trò là mask để lọc thông tin từ trạng thái cell.



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

**Hình 2.7: Cổng quên.**

**Cổng vào:** Cổng này dùng để cập nhật bộ nhớ với các thông tin mới. Ở đây có xuất hiện 2 hàm sigmoid và hàm tanh. Tác dụng của chúng cũng như trên. Output từ hàm sigmoid sẽ có tác dụng lọc thông tin đã qua xử lý từ output hàm tanh.

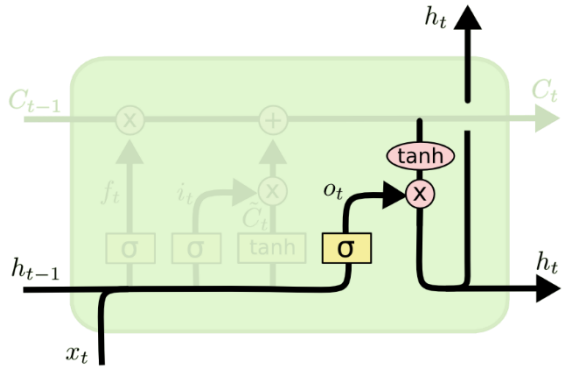


$$i_t = \sigma (W_i \cdot [h_{t-1}, x_t] + b_i)$$

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

**Hình 2.8: Cổng vào.**

Cổng ra: cổng này quyết định output của từ hiện tại là gì. Nó được lấy thông tin từ 2 nguồn: trạng thái cell và input hiện tại. Trạng thái cell sau khi chỉnh sửa sẽ đi qua hàm tanh và input hiện tại thì được đi qua hàm sigmoid. Kết hợp 2 kết quả trên để có được kết quả đầu ra. Kết quả đầu ra và cả trạng thái cell đều được đưa vào bước tiếp theo.



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

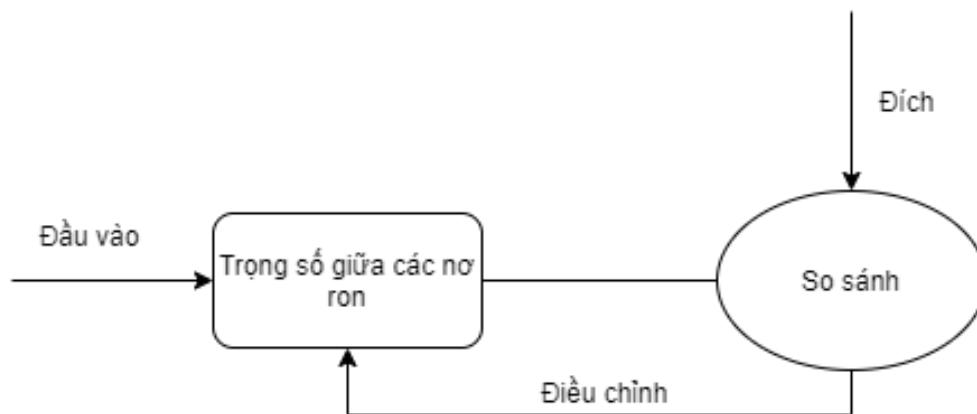
$$h_t = o_t * \tanh (C_t)$$

**Hình 2.9: Cổng ra.**

#### 2.1.4. Huấn luyện mạng nơ ron

Các mạng nơ ron được điều chỉnh hoặc huấn luyện để hướng các đầu vào riêng biệt đến các đích. Cấu trúc huấn luyện mạng được chỉ ra ở hình 2.10. Hàm trọng số của mạng được điều chỉnh dựa trên cơ sở so sánh đầu vào với đầu ra mong muốn cho tới khi đầu vào phù hợp với đích mong muốn. Những cặp đầu vào/đích được sử dụng để giá sát cho sự huấn luyện mạng.





**Hình 2.10: Cấu trúc huấn luyện mạng nơ ron.**

Để có được một số cặp vào/ra, ở đó mỗi giá trị vào được gửi đến mạng và giá trị ra tương ứng được thực hiện bằng mạng là sự xem xét và so sánh với giá trị mong muốn. Thông thường sẽ tại một sai số bởi lẽ giá trị mong muốn không hoàn toàn phù hợp với giá trị thực. Sau mỗi lần chạy, ta có tổng bình phương của tất cả các sai số. Sai số này được sử dụng để xác định các trọng số mới.

Sau mỗi lần chạy, trọng số của mạng được sửa đổi với đặc tính tốt hơn tương ứng với đặc tính mong muốn. Từng cặp giá trị vào/ra phải được kiểm tra và trọng số sẽ được điều chỉnh một vài lần. Sự thay đổi trọng số của mạng được dừng lại nếu tổng các bình phương sai số nhỏ hơn một giá trị đặt trước hoặc đã chạy đủ một số lần chạy xác định (trong trường hợp này mạng có thể không thoả mãn yêu cầu đặt ra do sai lệch còn cao).

## 2.2. Word Embedding

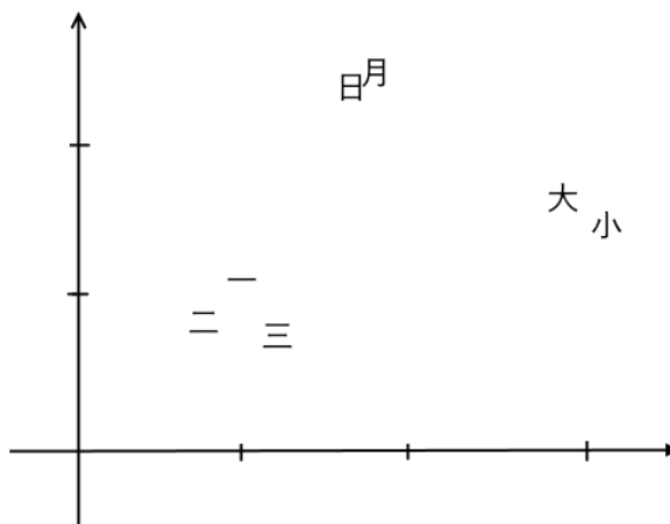
Đối với xử lý ngôn ngữ tự nhiên nói chung và dịch máy nói riêng, dữ liệu thường là dạng chuỗi ký tự. Con người nhìn chuỗi ký tự này và xử lý nội dung ở dạng các từ được ghép nối với nhau. Câu hỏi được đặt ra tương tự đối với máy tính. Làm thế nào để biểu diễn một chuỗi ký tự thành các con số để máy tính xử lý, đặc biệt trong các mô hình học máy khi mà dữ liệu đầu vào đóng vai trò cực kỳ quan trọng để xây dựng được mô hình hiệu quả. Một trong những cách biểu diễn tốt và được sử dụng phổ biến hiện nay đó là Word Embedding. Kỹ thuật trên cho phép biểu diễn mỗi token bằng một vector với số chiều thấp và có sự liên hệ ngữ nghĩa giữa các vector [10].

Ví dụ ta có câu “这只猫很大” - “con mèo rất to”, ở đây từ “大” có nghĩa là “to”, nếu ta thay từ này bằng từ “nhỏ” - “小”, câu trên sẽ là “con mèo rất nhỏ,” lúc này câu vẫn có nghĩa. Nhưng nếu ta thay bằng từ “đỏ” - “红”, lúc này câu sẽ không có nghĩa hoặc tối nghĩa. Bằng cách thay từ “大 - to” bằng từ “红 - đỏ” và nói cho mạng Neural biết rằng câu mới sinh ra là không hợp lệ, mạng Neural sẽ phải điều chỉnh các

tham số trong mạng của nó một cách hợp lý để đưa ra được đầu ra đúng như chúng ta mong muốn.

Hình 2.11 biểu diễn một cách biểu diễn Word Embedding đơn giản. Những từ có nghĩa tương tự nhau, có thể thay thế cho nhau sẽ được đặt gần nhau. Khoảng cách càng gần thì càng có khả năng thay thế hơn.

Như vậy, con người dịch thuật sử dụng ngôn ngữ tự nhiên, máy dịch sử dụng Word Embedding.



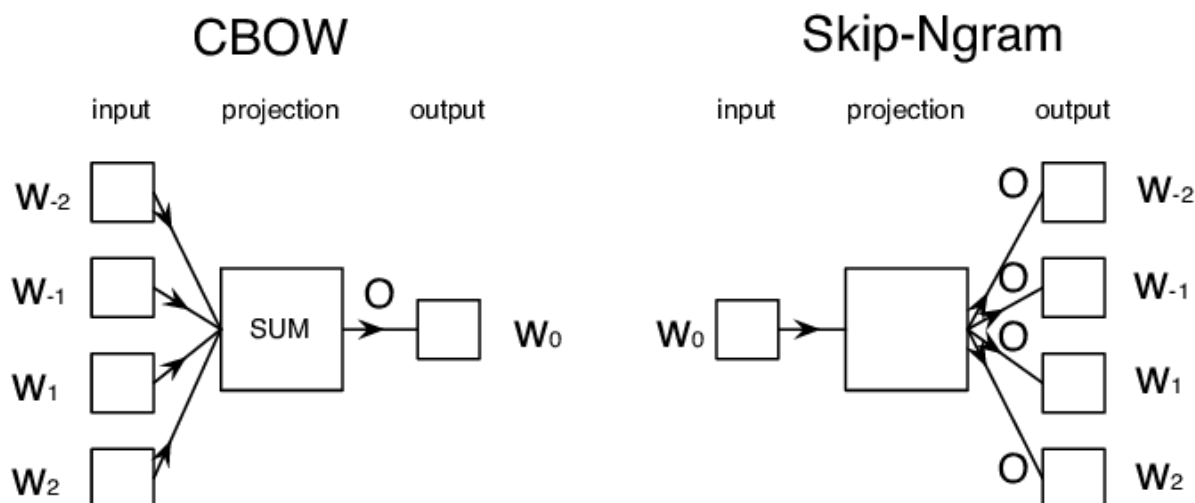
**Hình 2.11: Biểu diễn Word Embedding.**

### **2.2.1. Word2vec**

Word2Vec là 1 trong những mô hình đầu tiên về Word Embedding sử dụng mạng nơ-ron, vẫn khá phổ biến ở thời điểm hiện tại, có khả năng vector hóa từng từ dựa trên tập các từ chính và ngữ cảnh... Về mặt toán học, thực chất Word2Vec là việc ánh xạ từ từ 1 tập các từ (vocabulary) sang 1 không gian vector, mỗi vector được biểu diễn bởi n số thực. Mỗi từ ứng với 1 vector cố định. Sau quá trình huấn luyện mô hình bằng thuật toán backpropagation, trọng số các vector của từng từ được cập nhật liên tục. Từ đó, ta có thể thực hiện tính toán bằng các khoảng cách quen thuộc như euclidean, cosine, mahattan..., những từ càng “gần” nhau về mặt khoảng cách thường là các từ hay xuất hiện cùng nhau trong văn cảnh, các từ đồng nghĩa, các từ thuộc cùng 1 trường từ vựng...

Word2Vec bao gồm 2 cách tiếp cận chính, đó là CBOW và Skip-gram. Mô hình chung của Word2Vec (cả CBOW và Skip-gram) đều dựa trên một mạng nơ-ron khá đơn giản. Gọi  $V$  là tập các tất cả các từ hay vocabulary với  $n$  từ khác nhau. Layer input biểu diễn dưới dạng one-hot encoding với  $n$  node đại diện cho  $n$  từ trong vocabulary. Activation function (hàm kích hoạt) chỉ có tại layer cuối là softmax function, loss function là cross entropy loss, tương tự như cách biểu diễn mô hình của các bài toán

classification thông thường vậy. Ở giữa 2 layer input và output là 1 layer trung gian với size =  $k$ , chính là vector sẽ được sử dụng để biểu diễn các từ sau khi huấn luyện mô hình.



**Hình 2.12: Mô hình CBOW và Skip-grams.**

**CBOW:** ý tưởng chính của CBOW là dựa vào các context word (hay các từ xung quanh) để dự đoán center word (từ ở giữa). CBOW có điểm thuận lợi là training mô hình nhanh hơn so với mô hình skip-gram, thường cho kết quả tốt hơn với frequency words (hay các từ thường xuất hiện trong văn cảnh).

**Skip-gram:** ngược lại với CBOW, dùng target word để dự đoán các từ xung quanh. Skip-gram huấn luyện chậm hơn. Thường làm việc khá tốt với các tập dữ liệu nhỏ, đặc biệt do đặc trưng của mô hình nên khả năng vector hóa cho các từ ít xuất hiện tốt hơn CBOW.

Như vậy, bằng cách sử dụng hai mô hình trên, chúng ta có thể biểu diễn được bất kỳ từ nào thông qua vector số học chứa đựng ngữ nghĩa của từ và mối quan hệ với các từ khác, từ đó suy luận được ngữ cảnh và dự đoán được các từ lân cận.

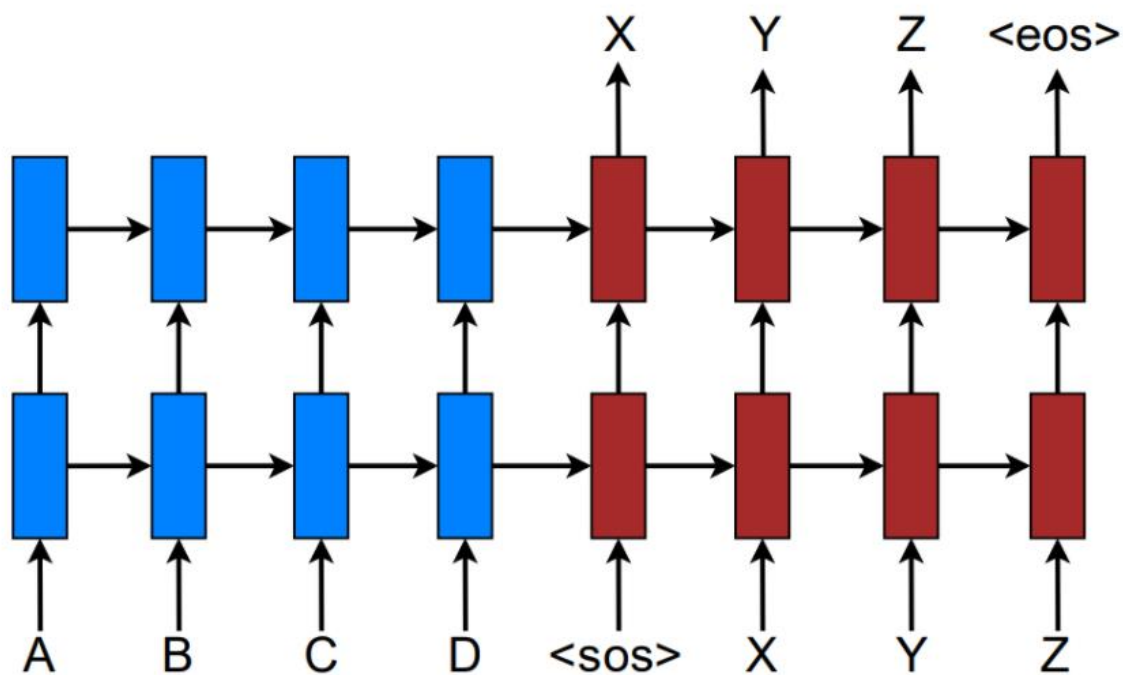
### 2.2.2. GloVe

GloVe cũng là một thuật toán nhằm tạo ra vector biểu diễn cho từ bằng cách giảm chiều vector từ trên ma trận đồng xuất hiện. Đầu tiên, thuật toán xây dựng một ma trận lớn chứa thông tin về tần suất của mỗi từ, được nhìn thấy trong các ngữ cảnh tương ứng. Số lượng ngữ cảnh là rất lớn, vì về cơ bản nó có kích thước tổ hợp. Sau đó nhân các ma trận này để tạo ra một ma trận các từ có chiều thấp hơn, trong đó mỗi ngữ cảnh mang lại một biểu diễn vector cho mỗi từ. Điều này đạt được bằng cách giảm thiểu tổn thất tái cấu trúc của người dùng, tìm kiếm các biểu diễn chiều thấp hơn có thể giải thích phương sai trong dữ liệu chiều cao.

## 2.3. Mô hình seq2seq

Đây là mô hình được đưa ra trong bài nghiên cứu Sequence to Sequence Learning with Neural Networks (Sutskever et al., 2014) [8] mà theo đó một câu sẽ được dịch bằng cách đưa vào một bộ mã hóa và nối tiếp với một bộ giải mã để dịch ra một câu ở ngôn ngữ khác.

Cụ thể, ngôn ngữ nguồn sẽ được mã hóa thành một vector và được đưa vào bộ mã hóa. Sau đó bộ giải mã sẽ lần lượt sinh từng từ trong chuỗi đầu ra dựa trên vector đầu vào và những từ được dự đoán trước đó cho tới khi gặp từ kết thúc câu. Trong mô hình seq2seq chúng ta có thể sử dụng những thành phần khác nhau cho bộ mã hóa và giải mã như RNN hoặc những cải tiến như LSTM và GRU để giải quyết vấn đề phụ thuộc xa và biểu diễn các mối quan hệ phụ thuộc vào ngữ cảnh của câu.



**Hình 2.13: Kiến trúc của mô hình Seq2Seq.**

Các thành phần chính của mô hình Sequence-to-Sequence bao gồm:

- Bộ Encoder được sử dụng để ánh xạ chuỗi token trong ngôn ngữ nguồn đầu vào thành một vector có kích thước cố định. Tại mỗi bước mã hóa, Encoder sẽ nhận vector tương ứng với mỗi token trong chuỗi đầu vào để tạo ra vector trạng thái ẩn s đại diện cho chuỗi đầu vào tại bước mã hóa cuối cùng.
- Bộ Decoder sử dụng vector s như khởi tạo cho trạng thái ẩn đầu tiên và tạo ra chuỗi các token ở ngôn ngữ đích tại mỗi bước giải mã. Do đó, hàm xác suất có điều kiện có thể được phân tích như sau:

$$p(y_1, \dots, y_m | x_1, \dots, x_n) = \prod_{j=1}^m p(y_j | s, \dots, y_{j-1})$$

Trong vế phải của công thức trên, mỗi phân bố  $p(y_j | s, \dots, y_{j-1})$  mô tả xác suất xuất hiện của token  $y_j$  với vector đại diện cho câu đầu vào  $s$  và các token trong chuỗi đầu ra đứng trước nó. Phân bố này được biểu diễn bằng một hàm softmax trên tất cả token trong tập từ vựng ở ngôn ngữ đích.

Công thức trên có thể được viết lại thành dạng như sau:

$$\log p(y|x) = \sum_{j=1}^m \log p(y_j | y_{<j}, s)$$

Mỗi token  $y_j$  có xác suất xuất hiện được tính như sau:

$$p(y_j | y_{<j}, s) = \text{softmax}(g(h_j))$$

Trong đó  $g$  là hàm dùng để biến đổi trạng thái ẩn  $h_j$  của bộ giải mã tại bước giải mã tương ứng thành vector có kích thước bằng kích thước của tập từ vựng trong ngôn ngữ đích. Trạng thái ẩn  $h_j$  được tính như sau:

$$h_j = f(h_{j-1}, s)$$

Trong đó  $f$  là hàm biểu diễn chung cho quá trình tính trạng thái ẩn tại bước hiện tại từ trạng thái ẩn đầu ra của bước trước bằng mạng RNN hoặc bằng những cải tiến khác như LSTM và GRU. Trong mô hình của tác giả Sutskever, vector  $s$  đại diện cho câu nguồn chỉ được sử dụng một lần để làm trạng thái ẩn đầu tiên cho bộ giải mã. Trong mô hình của tác giả Bahdanau và cộng sự,  $s$  là một vector đặc biệt được sử dụng xuyên suốt tại mỗi bước trong quá trình giải mã.

Hàm mất mát cần tối ưu hóa trong quá trình huấn luyện là một hàm có dạng tích của các hàm cross-entropy:

$$L = - \sum_{j=1}^m \sum_{i=1}^V q_{j,i} \log(p_{j,i})$$

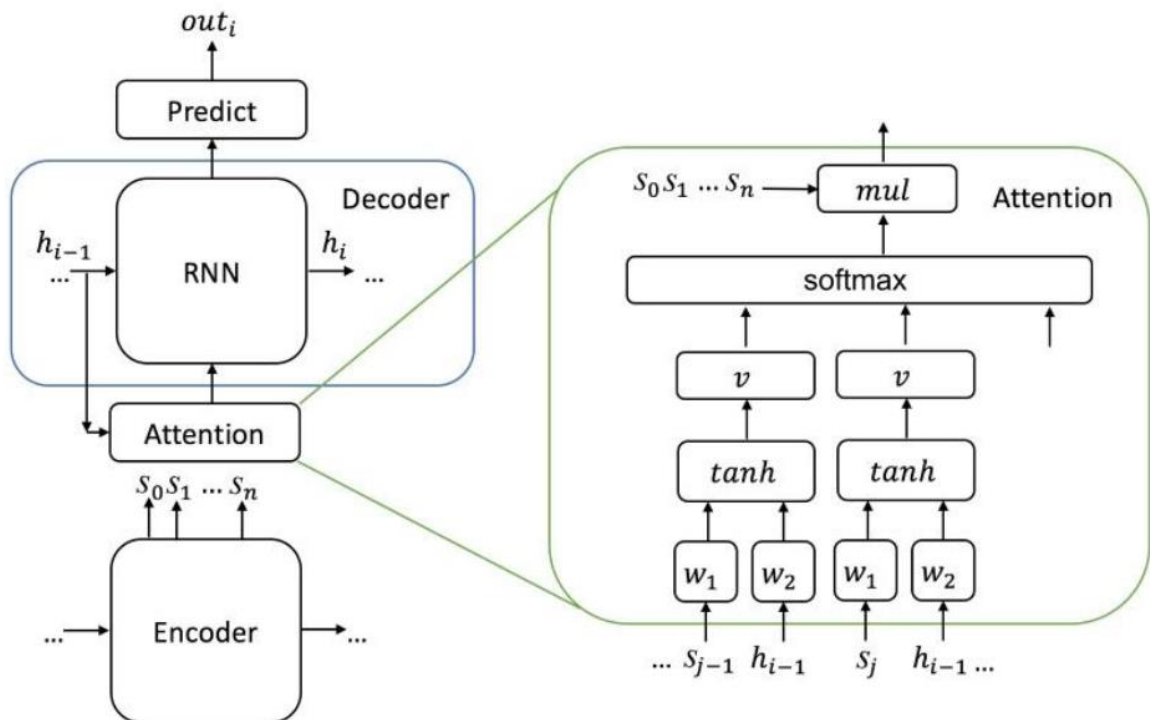
Trong đó,  $p_{j,i}$  là phần tử thứ  $i$  của vector one-hot  $q_j$  có kích thước  $V$  tại bước giải mã thứ  $j$ . Vector  $q_j$  biểu diễn cho token thứ  $j$  trong chuỗi token nhãn đầu ra từ tập huấn luyện.  $p_{j,i}$  là phần tử thứ  $i$  của vector  $p_j$  cũng có kích thước  $V$  với  $p_j = \text{softmax}(g(h_j))$ .

Về cơ bản sau khi quá trình huấn luyện hoàn tất, chúng ta sẽ tạo ra bản dịch  $\hat{y}$  từ một chuỗi đầu vào  $\hat{x}$  chưa biết trước bằng cách tính toán sinh ra bản dịch có khả năng xuất hiện cao nhất dựa vào mô hình thu được sau khi huấn luyện:

$$\hat{y} = \underset{y}{\operatorname{argmax}}(p(y|\hat{x}))$$

Mô hình seq2seq cơ bản có nhược điểm là yêu cầu RNN decoder sử dụng toàn bộ thông tin mã hóa từ chuỗi đầu vào cho dù chuỗi đó dài hay ngắn. Thứ hai, RNN encoder cần phải mã hóa chuỗi đầu vào thành một vec-tơ duy nhất và có độ dài cố định. Ràng buộc này không thực sự hiệu quả vì trong thực tế, việc sinh ra từ tại một bước thời gian trong chuỗi đầu ra có khi phụ thuộc nhiều hơn vào một số những thành phần nhất định trong chuỗi đầu vào. Ví dụ, khi dịch một câu từ tiếng nước này sang tiếng nước khác, chúng ta thường quan tâm nhiều hơn đến ngữ cảnh xung quanh từ hiện tại so với các từ khác trong câu. Kỹ thuật attention được đưa ra để giải quyết vấn đề đó.

Kỹ thuật attention được đưa ra lần đầu vào năm 2014 bởi Bahdanau và cộng sự trong công trình nghiên cứu về dịch máy. Ở mức trừu tượng, kỹ thuật attention nói lỏng điều kiện rằng toàn bộ chuỗi đầu vào được mã hóa bằng một vector duy nhất. Thay vào đó các từ trong chuỗi đầu vào sẽ được RNN encoder mã hóa thành một dãy các vector. Sau đó RNN decoder áp dụng kỹ thuật attention mềm dẻo (soft attention) bằng cách lấy tổng có trọng số của dãy các vector mã hóa. Các trọng số trong mô hình này được tính bằng một mạng neural truyền thẳng. RNN encoder, RNN decoder và các tham số trong kỹ thuật attention được huấn luyện đồng thời từ dữ liệu. Hình dưới đây minh họa mô hình seq2seq sử dụng attention trong bài toán dịch máy.



**Hình 2.14:** Minh họa mô hình seq2seq dung trong bài toán dịch máy.

## 2.4. Mô hình Transformer

### 2.4.1. Giới thiệu

RNN, LSTM là các phương pháp tiếp cận hiện đại trong mô hình ngôn ngữ và dịch máy. Từ đó, đã có nhiều nỗ lực cải tiến mô hình ngôn ngữ và kiến trúc mã hóa-giải mã.

Các mô hình hồi quy thường tính toán theo vị trí các ký tự các chuỗi đầu vào và đầu ra. Việc căn chỉnh vị trí này trong các bước tính toán, sẽ tạo ra các trạng thái ẩn  $h_t$ . Bản chất tuần tự vốn có này loại trừ tính đồng thời trong các mẫu huấn luyện, điều này trở nên quan trọng khi mà chuỗi dài hơn do các ràng buộc bộ nhớ bị giới hạn theo các mẫu ví dụ. Các nghiên cứu gần đây đã đạt được những cải tiến đáng kể về hiệu quả tính toán thông qua các thủ thuật và tính toán có điều kiện, đồng thời cải thiện hiệu suất của mô hình. Tuy nhiên, hạn chế của tính toán tuần tự vẫn còn.

Các cơ chế attention đã trở thành một phần không thể thiếu trong mô hình tuần tự, cho phép mô hình hóa các phụ thuộc mà không quan tâm đến khoảng cách của chuỗi đầu vào và đầu ra. Gần như trong tất cả các trường hợp, các cơ chế attention được sử dụng cùng với mạng hồi quy.

Dưới đây, mô hình Transformer được đề xuất, một kiến trúc mô hình tránh việc hồi quy mà thay vào đó là hoàn toàn dựa vào cơ chế attention để đưa ra sự phụ thuộc giữa đầu vào và đầu ra.

### 2.4.2. Self-attention

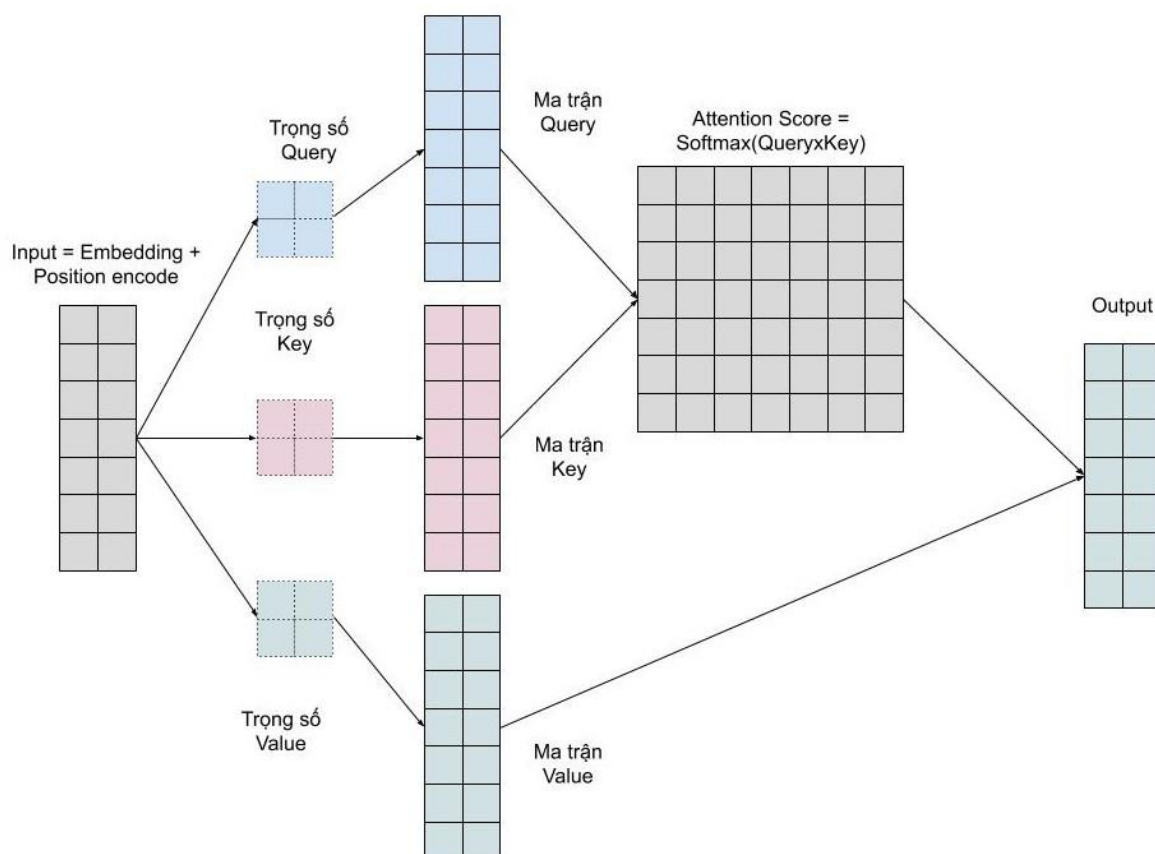
Trước khi đi chi tiết vào mô hình, tôi sẽ trình bày về self-attention – “trái tim” của mô hình transformer. Self-attention là cơ chế giúp Transformers “hiểu” được sự liên quan giữa các từ trong một câu. Có thể tưởng tượng self-attention giống như cơ chế tìm kiếm. Với một từ cho trước, cơ chế này sẽ cho phép mô hình tìm kiếm trong các từ còn lại để xác định từ nào liên quan để sau đó thông tin sẽ được mã hóa dựa trên tất cả các từ trên.

Đầu vào của self-attention là 3 vector query, key, value. Các vector này được tạo ra bằng cách nhân ma trận biểu diễn các từ đầu vào với ma trận học tương ứng.

- query vector là vector dùng để chứa thông tin của từ được tìm kiếm, so sánh.
- key vector là vector dùng để biểu diễn thông tin các từ được so sánh với từ cần tìm kiếm ở trên.
- value vector là vector biểu diễn nội dung, ý nghĩa của các từ.

Vector attention cho một từ thể hiện tính tương quan giữa 3 vector này được tạo ra bằng cách nhân tích vô hướng giữa chúng và sau đó được chuẩn hóa bằng hàm softmax. Cụ thể quá trình tính toán như sau:

- Bước 1: Tính ma trận query, key, value bằng cách nhân input với các ma trận trọng số tương ứng
- Bước 2: Nhân hai ma trận query, key vừa tính được với nhau với ý nghĩa so sánh giữa câu query và key để học mối tương quan. Sau đó các giá trị sẽ được chuẩn hóa về khoảng [0-1] bằng hàm softmax với ý nghĩa 1 khi câu query giống với key ngược lại, 0 có nghĩa là không giống
- Bước 3: Output sẽ được tính bằng cách nhân ma trận vừa được tạo ra ở bước 2 với ma trận value



**Hình 2.15: Quá trình tính toán vector attention.**

Hai hàm attention phổ biến được sử dụng là additive attention và dot-product attention. Một điểm đặc biệt dot-product attention là chia cho  $\sqrt{\text{Số chiều của vector key}}$ , additive attention tính toán sử dụng một mạng feed-forward với một tầng ẩn. Về mặt lý thuyết, cả hai giống nhau về độ phức tạp nhưng dot-product attention trong thực tế nhanh hơn và hiệu quả hơn vì có thể sử dụng thuật toán nhân ma trận tối ưu. Trong đồ án sẽ sử dụng dot-product attention

Để hiểu rõ hơn về cách tính toán chúng ta sẽ thực hiện tính toán vector attention qua các bước như sau: (1) Chuẩn bị đầu vào, (2) Khởi tạo trọng số, (3) Xác định key, query và value, (4) Tính điểm số attention với đầu vào 1, (5) Tính giá trị hàm softmax,



(6) Nhân điểm số trên với value, (7) Tính đầu ra cho đầu vào 1, (8) Lặp lại các bước từ 4 đến 7 với các đầu vào khác. Cụ thể như sau:

Trong ví dụ này, chúng ta sẽ thực hiện tính toán với 3 đầu vào với số chiều là 4.

Input 1: [1, 0, 1, 0]

Input 2: [0, 2, 0, 2]

Input 3: [1, 1, 1, 1]

Mỗi một đầu vào phải có 3 biểu diễn đó là key, query và value. Giả sử chúng ta muốn các vector này có số chiều là 3, trong khi đầu vào có kích thước là 4. Do đó, một tập hợp trọng số phải là ma trận có kích thước là  $4 \times 3$ . Chúng ta sẽ khởi tạo bộ trọng số như sau:

Trọng số key

[[0, 0, 1],  
[1, 1, 0],  
[0, 1, 0],  
[1, 1, 0]]

Trọng số query

[[1, 0, 1],  
[1, 0, 0],  
[0, 0, 1],  
[0, 1, 1]]

Trọng số value

[[0, 2, 0],  
[0, 3, 0],  
[1, 0, 3],  
[1, 1, 0]]

Lưu ý rằng trong cài đặt mạng nơ ron, các trọng số này thường là các số rất nhỏ, được khởi tạo ngẫu nhiên bằng cách sử dụng các phân phối ngẫu nhiên như Gaussian, Xavier và Kaimin. Khởi tạo này sẽ được thực hiện một lần trước khi huấn luyện.

Bây giờ chúng ta đã có 3 bộ trọng số, chúng ta sẽ tính các ma trận biểu diễn key, query và value cho các đầu vào.

Ma trận key cho đầu vào 1:

$$\begin{array}{r}
[0, 0, 1] \\
[1, 0, 1, 0] \times [1, 1, 0] = [0, 1, 1] \\
[0, 1, 0] \\
[1, 1, 0]
\end{array}$$

Tính toán tương tự cho đầu vào 2 và đầu vào 3 ta có:

$$\begin{array}{r}
[0, 0, 1] \\
[0, 2, 0, 2] \times [1, 1, 0] = [4, 4, 0] \\
[0, 1, 0] \\
[1, 1, 0]
\end{array}$$

$$\begin{array}{r}
[0, 0, 1] \\
[1, 1, 1, 1] \times [1, 1, 0] = [2, 3, 1] \\
[0, 1, 0] \\
[1, 1, 0]
\end{array}$$

Để tính toán nhanh hơn, chúng ta có thể tính toán ma trận key như sau:

$$\begin{array}{r}
[0, 0, 1] \\
\begin{array}{ccc}
[1, 0, 1, 0] & [1, 1, 0] & [0, 1, 1] \\
[0, 2, 0, 2] \times [0, 1, 0] & = & [4, 4, 0] \\
[1, 1, 1, 1] & [1, 1, 0] & [2, 3, 1]
\end{array}
\end{array}$$

Tính toán tương tự với ma trận value:

$$\begin{array}{r}
[0, 2, 0] \\
\begin{array}{ccc}
[1, 0, 1, 0] & [0, 3, 0] & [1, 2, 3] \\
[0, 2, 0, 2] \times [1, 0, 3] & = & [2, 8, 0] \\
[1, 1, 1, 1] & [1, 1, 0] & [2, 6, 3]
\end{array}
\end{array}$$

và ma trận query:

$$\begin{array}{r}
[1, 0, 1] \\
\begin{array}{ccc}
[1, 0, 1, 0] & [1, 0, 0] & [1, 0, 2] \\
[0, 2, 0, 2] \times [0, 0, 1] & = & [2, 2, 2] \\
[1, 1, 1, 1] & [0, 1, 1] & [2, 1, 3]
\end{array}
\end{array}$$

Tiếp đến chúng ta sẽ tính toán điểm số attention với từng đầu vào. Giá trị này sẽ được tính bằng cách nhân ma trận query của từng đầu vào với ma trận chuyển vị của ma trận value. Cụ thể như sau:

$$\begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} \times \begin{bmatrix} 0 & 4 & 2 \\ 1 & 4 & 3 \\ 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 4 \end{bmatrix}$$

Tiếp đến chúng ta sẽ chuẩn hóa giá trị trên bằng hàm softmax:

$$\text{softmax}([2, 4, 4]) = [0, 0, 0.5, 0.5]$$

Sau đó, chúng ta sẽ nhân giá trị hàm softmax với ma trận value:

$$\begin{aligned} 1: & 0.0 * [1, 2, 3] = [0.0, 0.0, 0.0] \\ 2: & 0.5 * [2, 8, 0] = [1.0, 4.0, 0.0] \\ 3: & 0.5 * [2, 6, 3] = [1.0, 3.0, 1.5] \end{aligned}$$

Các giá trị trên sẽ được cộng với nhau để thu được ma trận attention

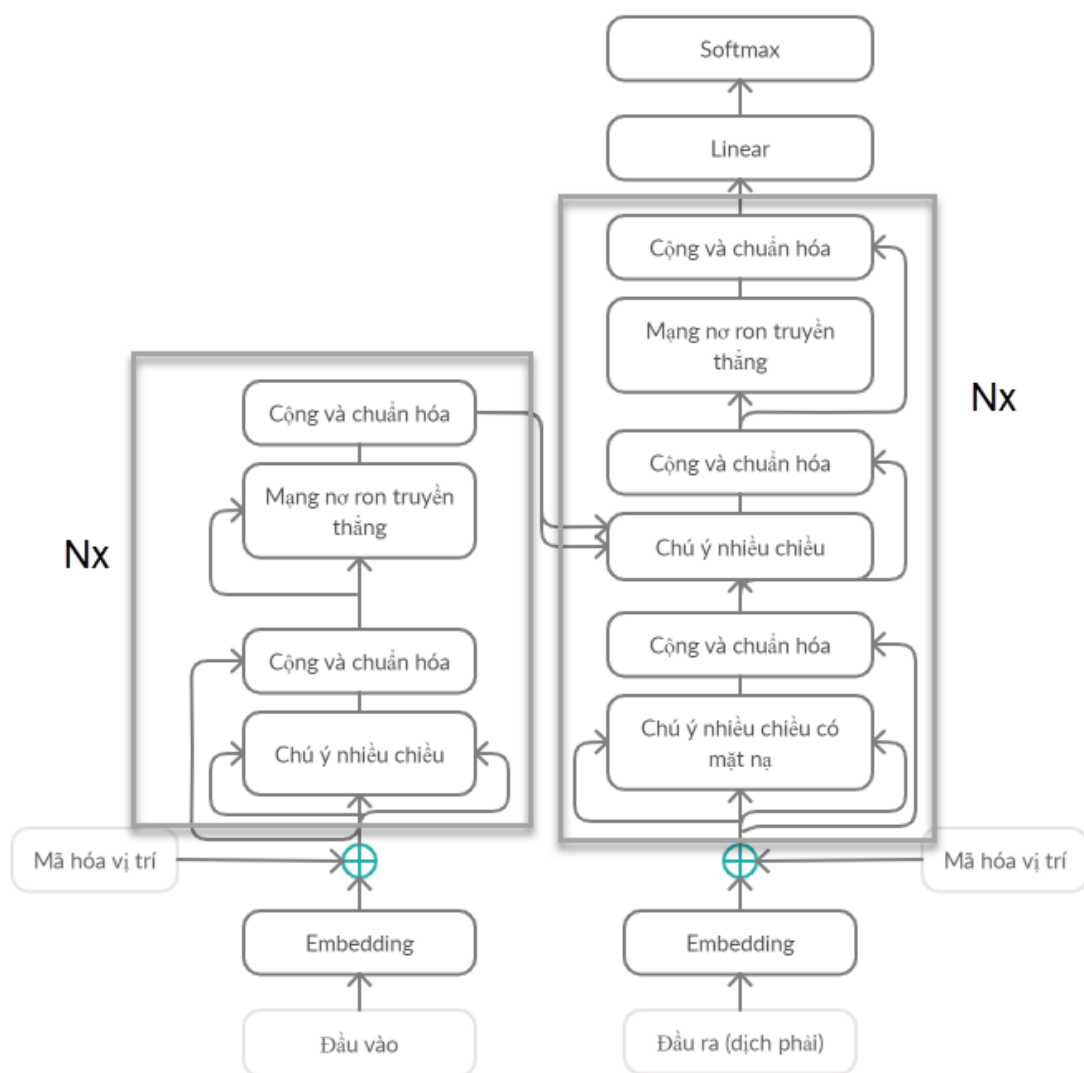
$$\begin{aligned} & [0.0, 0.0, 0.0] \\ & + [1.0, 4.0, 0.0] \\ & + [1.0, 3.0, 1.5] \\ & \text{-----} \\ & = [2.0, 7.0, 1.5] \end{aligned}$$

Ma trận  $[2.0, 7.0, 1.5]$  chính là biểu diễn attention của đầu vào 1, nó thể hiện mối quan hệ với tất cả các key khác và cả chính nó. Thực hiện tương tự với đầu vào 2 và 3, ta có ma trận attention với các đầu vào như sau:

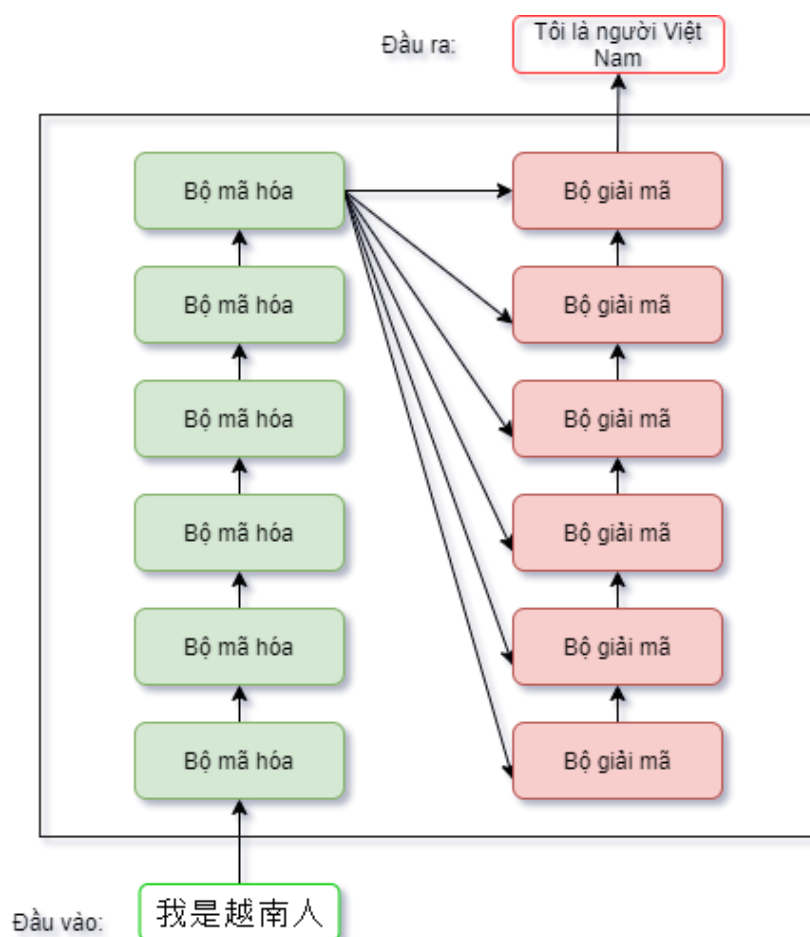
$$\begin{aligned} & [[2.0, 7.0, 1.5], \text{ \# attention 1} \\ & [2.0, 8.0, 0.0], \text{ \# attention 2} \\ & [2.0, 7.8, 0.3]] \text{ \# attention 3} \end{aligned}$$

### 2.4.3. Tổng quan mô hình

Đây là mô hình được các kỹ sư của Google giới thiệu vào năm 2017 trong bài báo Attention Is All You Need [2]. Cũng giống như các mô hình dịch máy khác, mô hình transformer cũng bao gồm hai phần lớn là bộ mã hóa và bộ giải mã. Bộ mã hóa biểu diễn ngôn ngữ nguồn thành các vector, bộ giải mã sẽ nhận các vector biểu diễn này và dịch nó sang ngôn ngữ đích. Chi tiết các thành phần của bộ mã hóa và giải mã được thể hiện như hình 2.16, bộ mã hóa và giải mã lần lượt nằm ở cột bên trái và bên phải của hình vẽ.



**Hình 2.16: Kiến trúc mô hình Transformer.**



**Hình 2.17: Bộ mã hóa và giải mã trong mô hình transformer.**

Một trong những ưu điểm của transformer là mô hình có khả năng xử lý song song cho các từ. Đầu vào sẽ được đẩy vào cùng một lúc. Bộ mã hóa của mô hình transformer bao gồm một tập gồm  $N = 6$  lớp giống nhau, mỗi lớp bao gồm 2 lớp con. Lớp đầu tiên là cơ chế multi-head self-attention, và lớp thứ 2 là mạng feed-forward kết nối đầy đủ. Đầu ra của mỗi lớp con là  $\text{LayerNorm}(x + \text{Sublayer}(x))$ , trong đó  $\text{Sublayer}(x)$  là một hàm được thực hiện bởi chính lớp con đó.

Bộ giải mã: cũng bao gồm tập gồm  $N = 6$  lớp giống nhau. Ngoài hai lớp con giống như bộ mã hóa, bộ giải mã còn có một lớp để thực hiện multi-head attention trên đầu ra của lớp giải mã. Ở đây sẽ có thay đổi cơ chế self-attention trong bộ mã hóa. Dưới đây sẽ trình bày chi tiết về bộ mã hóa và giải mã của mô hình transformer.

#### 2.4.4. Bộ mã hóa

Dữ liệu đầu vào sẽ được mã hóa thành các vector, sau đó sẽ được đưa vào các lớp được xếp chồng lên nhau. Các thành phần của một lớp được biểu diễn như hình 2.18.



**Hình 2.18: Một lớp trong bộ mã hóa của mô hình Transformer.**

#### 2.4.4.1. Input Embedding

Các câu đầu vào sẽ được mã hóa thành các vector bằng việc sử dụng Word Embedding được trình bày ở phần 2.2

#### 2.4.4.2. Positional Encoding

Input embeddings phần nào cho giúp ta biểu diễn ngữ nghĩa của một từ, tuy nhiên cùng một từ ở vị trí khác nhau của câu lại mang ý nghĩa khác nhau. Đó là lý do Transformers có thêm một phần Positional Encoding để cho biết thêm thông tin về vị trí của một từ. Giá trị này được tính như sau:

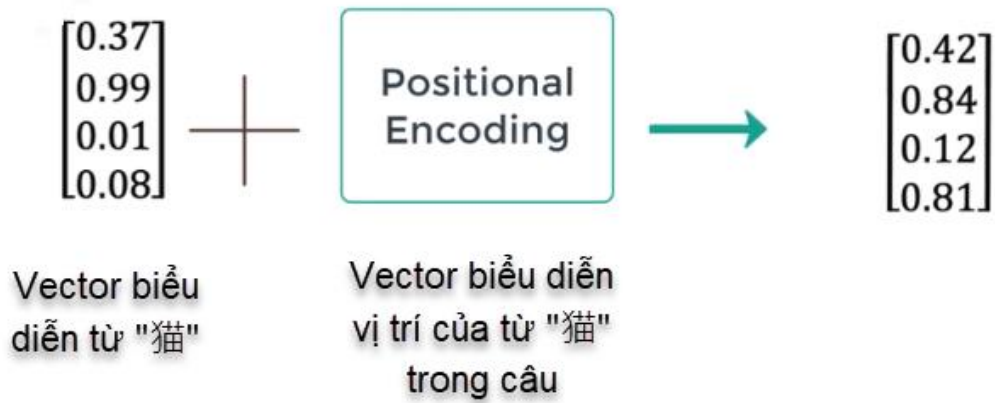
$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}})$$

Trong đó:

- pos là vị trí của từ trong câu
- PE là giá trị phần tử thứ i trong embeddings có độ dài  $d_{model}$

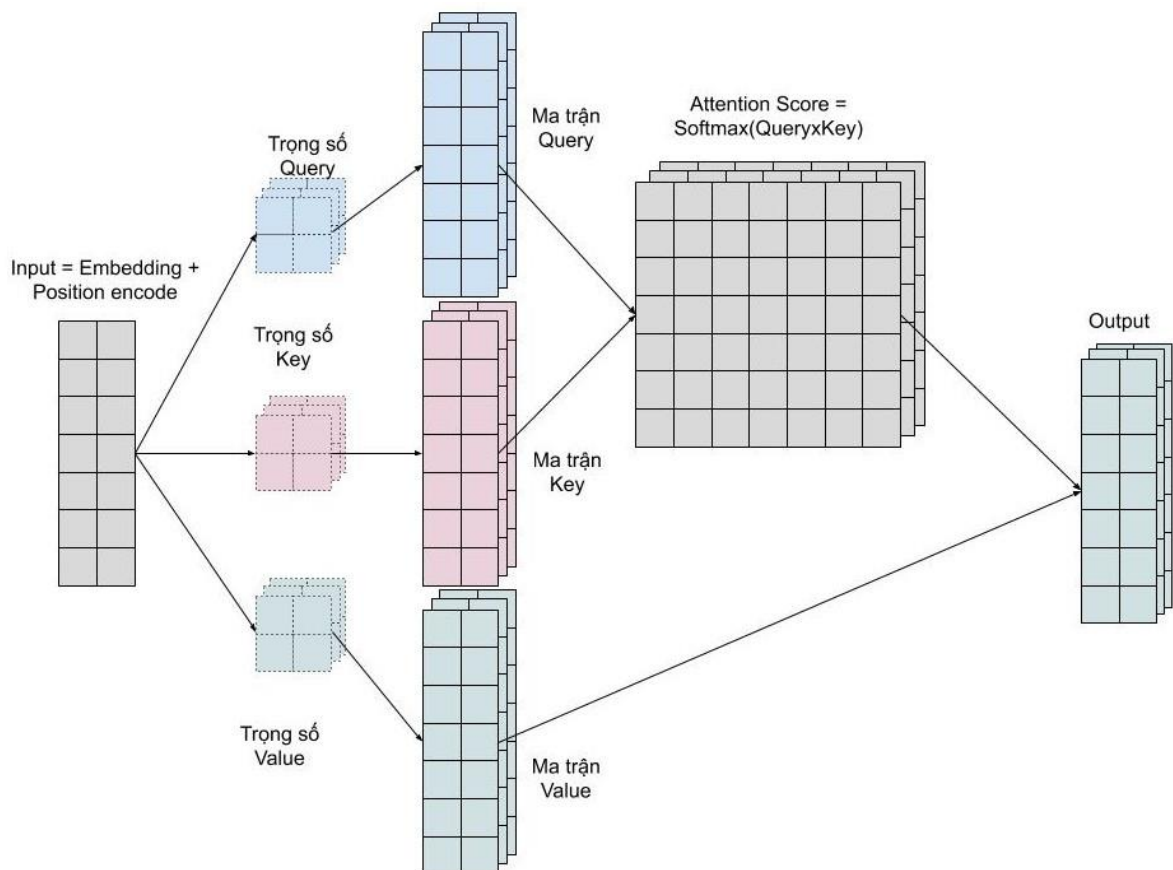
Như vậy bộ mã hóa sẽ nhận ma trận biểu diễn của các từ đã được cộng với thông tin vị trí thông qua positional encoding.



**Hình 2.19: Ví dụ biểu diễn từ đầu vào.**

Sau đó, ma trận này sẽ được xử lý bởi Multi Head Attention. Multi Head Attention thực chất là sử dụng nhiều self-attention

#### 2.4.4.3. Multi Head Attention



**Hình 2.20: Quá trình tính toán vector attention với nhiều “head”.**

Vấn đề của self-attention là attention của một từ sẽ luôn “chú ý” vào chính nó. Chúng ta muốn mô hình có thể học nhiều kiểu mối quan hệ giữa các từ với nhau. Ý tưởng là thay vì sử dụng một self-attention thì chúng ta sẽ sử dụng nhiều self-attention.

Đơn giản là cần nhiều ma trận query, key, value. Mỗi “head” sẽ cho ra output riêng, các ma trận này sẽ được kết hợp với nhau và nhân với ma trận trọng số để có được ma trận attention duy nhất.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$$

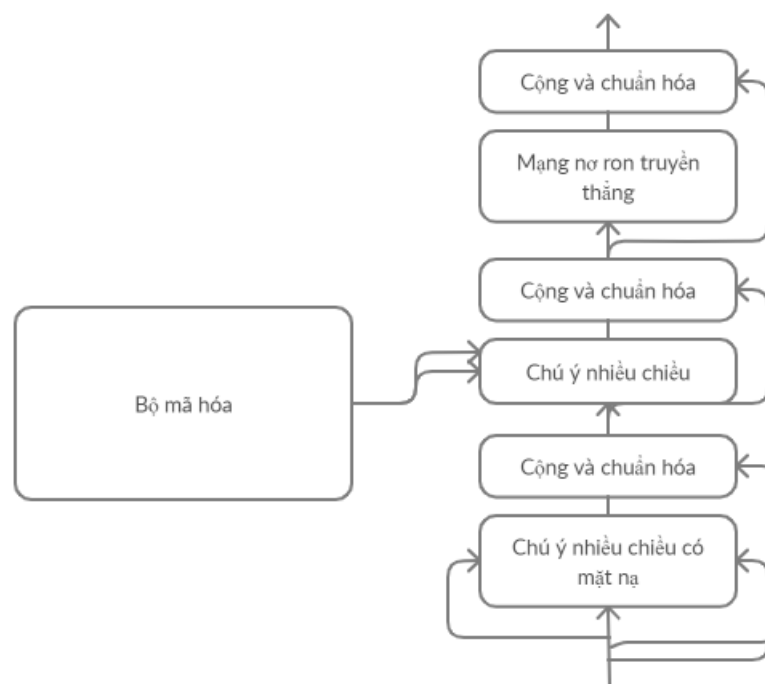
Mỗi encoder và decoder trong Transformer sử dụng N attention. Mỗi attention sẽ biến đổi tuyến tính q, k, k với một ma trận có thể huấn luyện khác nhau tương ứng.

Mỗi phép biến đổi cung cấp cho chúng ta một phép chiếu khác nhau cho q, k và v. Vì vậy, N attention cho phép xem mức độ phù hợp từ N quan điểm khác nhau. Điều này cuối cùng đẩy độ chính xác tổng thể cao hơn, ít nhất là theo kinh nghiệm.

Việc chuyển đổi cũng làm giảm kích thước đầu ra của chúng, do đó, thậm chí N attention được sử dụng, độ phức tạp tính toán vẫn giữ nguyên. Trong multi-head attention, ghép các vectơ đầu ra theo sau là một phép biến đổi tuyến tính.

#### 2.4.5. Bộ giải mã

Bộ giải mã thực hiện chức năng giải mã vector của câu nguồn thành câu đích, do đó bộ giải mã sẽ nhận thông tin từ bộ mã hóa là 2 vector key và value. Kiến trúc của bộ giải mã rất giống với bộ mã hóa, ngoại trừ có thêm một masked multi-head attention nằm ở giữ dùng để học mối liên quan giữ từ đang được dịch với các từ được ở câu nguồn.



**Hình 2.21: Bộ giải mã của mô hình transformer.**



Masked multi-head attention tất nhiên là multi-head attention mà chúng ta đã nói đến ở trên, tuy nhiên đi các từ ở tương lai chưa được mô hình dịch đến được che lại.

Trong bộ giải mã còn có một multi-head attention khác có chức năng chú ý các từ ở bộ mã hóa, layer này nhận vector key và value từ bộ mã hóa, và output từ layer phía dưới. Đơn giản bởi vì chúng ta muốn so sánh sự tương quan giữa từ đang được dịch với các từ nguồn.

#### **2.4.6. Ứng dụng Attention trong mô hình Transformer**

Mô hình Transformer sử dụng multi-head attention theo 3 cách khác nhau [2]. Thứ nhất là Trong lớp “encoder-decoder attention”, câu truy vấn đến từ lớp giải mã trước đó và các khóa và giá trị đến từ đầu ra của bộ mã hóa. Điều này cho phép tất cả các vị trí trong bộ giải mã sẽ tham gia vào tất cả các vị trí trong chuỗi đầu vào. Nó tương tự như cơ chế encoder-decoder attention trong mô hình sequence-to-sequence.

Tiếp đến là bộ mã hóa chứa các lớp self-attention. Trong một lớp self-attention, tất cả các khóa, giá trị và truy vấn đều đến từ cùng một nơi, trong trường hợp này đó là đầu ra của lớp trước đó trong mã hóa. Mỗi vị trí trong bộ mã hóa có thể tham gia vào tất cả các vị trí trong lớp trước của bộ mã hóa.

Ngoài ra, các lớp self-attention trong bộ giải mã cho phép mỗi vị trí trong bộ giải mã tham dự tất cả các vị trí trong bộ giải mã bao gồm cả vị trí của nó. Chúng ta cần ngăn chặn luồng thông tin bên trái trong bộ giải mã để bảo toàn thuộc tính tự động hồi quy. Điều này được thực hiện bên trong scaled dot-product attention bằng cách che đi tất cả các giá trị trong đầu vào của softmax tương ứng với các kết nối không hợp lệ.

# CHƯƠNG 3: DỊCH MÁY TRUNG-VIỆT DỰA VÀO MÔ HÌNH TRANSFORMER

## 3.1. Giới thiệu

Dịch máy là một trong những vấn đề khó và lâu đời nhất trong trí tuệ nhân tạo. Mặc dù có lịch sử lâu đời nhưng vấn đề dịch động hiện nay vẫn còn nhiều thách thức do vấn đề về mặt ngôn ngữ và văn hóa. Luận văn sẽ tập trung vào việc nghiên cứu dịch thuật với cặp ngôn ngữ Trung – Việt.

## 3.2. Định hướng giải pháp

Để xây dựng hệ thống dịch máy Trung-Việt, ở đây chúng ta sẽ sử dụng cách tiếp cận dịch máy mạng nơ ron, sử dụng mô hình Transformer như đã trình bày ở chương 2. Đầu tiên, hệ thống sẽ được “học” để có được một mạng nơ ron với bộ trọng số tương ứng. Đầu vào sẽ là một tập dữ liệu tiếng Trung và bản dịch tiếng Việt tương ứng. Dữ liệu này sẽ được tiền xử lý, sau đó đưa vào huấn luyện. Quá trình dịch sẽ sử dụng những thông tin đã “học” được ở trên để xây dựng câu dịch.

## 3.3. Thử nghiệm

### 3.3.1. Thử nghiệm mô hình Transformer

#### 3.3.1.1. Cấu hình phần cứng

**Bảng 3-1: Thông tin cấu hình phần cứng**

Cấu hình	Chỉ số
CPU	Intel(R) Xeon(R) CPU @ 2.30GHz
GPU	NVIDIA Tesla K80
RAM	16
OS	Linux

#### 3.3.1.2. Dữ liệu sử dụng

Dữ liệu tiếng Trung được lấy từ tập truyện “全职高手” tại <http://www.jjwxc.net/>, bản dịch tiếng Việt được nhóm dịch giả của nhà Xuất bản Hà Nội dịch. Dữ liệu bao gồm 16.483 cặp câu Trung-Việt, được chia thành các tập dữ liệu như sau:

**Bảng 3-2: Thống kê về dữ liệu sử dụng**

	Số câu	Số tokens	Số từ	Độ dài trung bình câu
train.vi	14896	231789	7586	15.56049
train.cn	14896	234010	15104	15.70959
dev.vi	1070	17997	2662	16.81963
dev.cn	1070	17948	3724	16.77383
test.vi	517	8287	1741	16.02901
test.cn	517	8375	2253	16.19923

Dữ liệu sau khi được thu thập về sẽ được tách từ, sử dụng công cụ VnCoreNLP [12] với tiếng Việt và Stanford Word Segmenter [9] với tiếng Trung và được biểu diễn bằng các vector để đưa vào huấn luyện. Quá trình huấn luyện sẽ sử dụng Tensor2Tensor.

### 3.3.1.3. Tối ưu hóa

Có nhiều thuật toán tối ưu hóa có thể sử dụng, nổi bật hơn cả là và SGD. Mặc dù Adam mới được ra mắt gần đây và được cộng đồng nghiên cứu NLP sử dụng thường xuyên vì sự vượt trội rõ ràng của Adam so với SGD. Ở đây chúng ta sẽ sử dụng Adam [6].

Tuy nhiên learning rate sẽ được điều chỉnh trong suốt quá trình học, ý tưởng là sẽ khởi động tốc độ học ban đầu và giảm dần khi về cuối theo công thức sau:

$$lrate = d_{model}^{-0.5} * \min(step\_num^{-0.5}, step\_num * warmup\_steps^{-1.5})$$

Ngoài ra việc điều chỉnh các siêu tham số có tác động rất lớn tới độ phức tạp và độ nặng của mô hình, ở đây lựa chọn bộ tham số như sau:

- Mô hình: T2T cung cấp hai mô hình với hai bộ tham số đã được xác định trước là transformer\_big\_single\_gpu (BIG) và transformer\_base\_single\_gpu (BASE) mà khác biệt chính là ở kích thước của mô hình. Theo bài báo “Attention Is All You Need” [2], mô hình BIG tỏ ra tốt hơn mô hình BASE, mặc dù BIG tốn nhiều bộ nhớ hơn. Ở đây chúng ta sẽ lựa chọn mô hình BIG
- Kích thước lô (batch size): một tập dữ liệu huấn luyện có thể chia nhỏ thành các batch, mỗi một batch sẽ chứa các training samples, số lượng các samples

này được gọi là batch size. Việc lựa chọn batch size lớn hay nhỏ sẽ ảnh hưởng đến tốc độ tính toán và thông lượng đào tạo. Tốc độ tính toán sẽ giảm dần khi tăng dần batch size bởi không phải tất cả hoạt động của các GPU đều hoạt động song song theo lô. Ngược lại, thông lượng đào tạo tăng tuyến tính với kích thước của lô [6]. Ở đây sẽ lựa chọn batch size bằng 2000.

- Bộ mã hóa và giải mã là tổng hợp xếp chồng lên nhau của 6 layer. Mỗi layer bao gồm 2 layer con (sub-layer) trong đó sub-layer đầu tiên là multi-head self-attention với số head là 16. Sub-layer thứ hai là feedforward network. Đầu ra của mỗi sub-layer này sẽ có số chiều là 1024

**Bảng 3-3: Các tham số huấn luyện mô hình Transformer**

Tham số	Giá trị
N	6
hidden size	1024
batch size	2000
num head	16
optimizer	adam
warmup_steps	16.000
train steps	300.000

### 3.3.2. Thử nghiệm mô hình dịch máy nơ ron sử dụng RNN và Attention

Để so sánh với mô hình Transformer, tôi sẽ tiến hành thử nghiệm hệ thống dịch sử dụng RNN và Attention trên cùng một cấu hình phần cứng và dữ liệu sử dụng với mô hình Transformer. Để huấn luyện hệ dịch, tôi sử dụng công cụ OpenNMT [11], sử dụng các tham số theo bài báo của Sennrich và cộng sự [14]. Cụ thể như sau:

**Bảng 3-4: Các tham số huấn luyện sử dụng RNN và Attention**

Tham số	Giá trị
hidden layer size	1024
embedding size	512
encoder depth	2

Tham số	Giá trị
decoder depth	2
hidden dropout	0.2
embedding dropout	0.2
source word dropout	0.2
label smoothing	0.1
optimizer	adam
learning_rate	0.0001
warmup_scheme	16.000
infer_mode	300.000

### 3.4. Đánh giá

#### 3.4.1. Phương pháp đánh giá

Bilingual Evaluation Understudy Score hay ngắn gọn là BLEU score là một thang điểm được dùng phổ biến trong đánh giá chất lượng dịch máy. BLEU được Kishore Papineni và cộng sự đề xuất lần đầu vào năm 2002 qua bài nghiên cứu “A Method for Automatic Evaluation of Machine Translation” [4].

BLEU được tính dựa trên số lượng n-grams giống nhau giữa câu dịch của mô hình (output) với các câu tham chiếu tương ứng (label) có xét tới yếu tố độ dài của câu.

Số n-grams tối đa của BLEU là không giới hạn, nhưng vì xét về ý nghĩa, cụm từ quá dài thường không có nhiều ý nghĩa, và nghiên cứu cũng đã cho thấy là với 4-gram, điểm số BLEU trung bình cho khả năng dịch thuật của con người cũng đã giảm khá nhiều nên n-grams tối đa thường được sử dụng là 4-gram. Công thức để tính điểm đánh giá như sau:

$$score = \exp \left\{ \sum_{i=1}^N w_i \log(p_i) - \max \left( \frac{L_{ref}}{L_{tra}} - 1, 0 \right) \right\}$$

$$p_i = \frac{\sum_j NR_j}{\sum_j NT_j}$$

Trong đó:

- $NR_j$  là số lượng các n-grams trong phân đoạn j của bản dịch dùng để tham khảo
- $NT_j$  là số lượng các n-grams trong phân đoạn j của bản dịch bằng máy.
- $L_{ref}$  là số lượng các n-grams trong phân đoạn j của bản dịch bằng máy.
- $L_{tra}$  là số lượng các từ trong bản dịch bằng máy.

Giá trị score đánh giá mức độ tương ứng giữa hai bản dịch và nó được thực hiện trên từng phân đoạn, ở đây phân đoạn được hiểu là đơn vị tối thiểu trong các bản dịch, thông thường mỗi phân đoạn là một câu hoặc một đoạn. Việc thống kê đồ trùng khớp của các n-grams dựa trên tập hợp các ngrams trên các phân đoạn, trước hết là nó được tính trên từng phân đoạn, sau đó tính lại giá trị này trên tất cả các phân đoạn.

### 3.4.2. Kết quả

Dưới đây là điểm số BLEU khi sử dụng hệ thống để dịch bộ dữ liệu kiểm thử từ tiếng Trung sang tiếng Việt. Để đánh giá hệ thống, tôi đã so sánh với hệ thống dịch sử dụng RNN và attention với chung một bộ dữ liệu huấn luyện và kiểm thử. Có thể thấy rằng mô hình Transformer cho kết quả tốt hơn mô hình dịch nơ ron sử dụng RNN và attention.

**Bảng 3-5: Điểm BLEU của hệ thống dịch máy Trung – Việt**

Loại dịch máy	BLEU
RNN + Attention	16.73
Transformer	30.07

Dưới đây là một số kết quả dịch Trung Việt sử dụng mô hình Transformer:

**Bảng 3-6: Một số kết quả dịch**

STT	Đầu vào	Đầu ra	Tham chiếu
1	这里是一间很宽大的会议室，刚入门便可看到一面几乎占满整面墙壁的电子显示屏，上面显示得是 “ 荣耀职业联盟	Ở đây là một phòng họp rất lớn, khi mới bắt đầu, bạn có thể thấy một màn hình điện tử bao phủ gần như toàn bộ bức tường, hiển thị thứ hạng của “Liên minh chuyên nghiệp Vinh Quang” và	Nơi này là một phòng họp rất rộng , mới vào cửa đã có thể nhìn thấy ngay một tấm bảng điện tử chiếm trọn vách tường , mặt trên hiển thị bảng xếp hạng chiến tích của “ Liên

	” 的战绩排行和一些技术统计。	thống kê kỹ thuật.	minh chuyên nghiệp Vinh Quang ” và một ít số liệu thống kê .
2	现如今的荣耀已经发展到如此程度了吗？	Hiện nay Vinh Quang đã phát triển đến trình độ như vậy ư?	Vinh Quang giờ đây đã phát triển tới trình độ nào rồi ?
3	叶修摇了摇头，他心中并不太肯定。	Diệp Tu lắc đầu, trong lòng gã cũng không quá chắc chắn	Diệp Tu lắc đầu, trong lòng hắn không hề chắc chắn.
4	叶修这话说完，那三位已经都冲出去了，枫桦却还是望着那两个在喝酒的流氓茫然：我怎么过去那个石窗？	Diệp Tu nói xong, ba người kia đều lao ra, Phong Hoa vẫn nhìn hai tên lưu manh đang uống rượu: Tôi làm sao lại có thể qua được cửa sổ đá?	Diệp Tu vừa dứt lời, ba người đã cùng lao ra, Phong Hoa vẫn mờ mịt nhìn hai tên lưu manh đang rượu chè kia : Sao tôi qua được cái cửa sổ đá kia ?
5	另三人一怔，这个流木的加入他们一点反应都没有，他们还在等着离恨剑出现呢！	Ba người ngẩn ra , Lưu Mộc gia nhập bọn họ cũng không hề phản ứng, bọn họ vẫn đang chờ Ly Hận Kiếm xuất hiện!	Ba người khác thì ngẩn ra , họ chưa kịp có phản ứng khi Lưu Mộc gia nhập, cả đám còn đang chờ Ly Hận Kiếm xuất hiện đây!
6	但这一次，他的连击却没有那么顺畅，因为这已是狂暴状态下的猫妖，提速后的移动和攻击让月中眠的动作都有些跟不上了。	Nhưng lần này, gã đã không thuận theo , bởi vì người mà vẫn không làm trạng thái Bá Thể, đã là trạng thái cuồng bạo để công kích vào trạng thái, tăng tốc và công kích của Nguyệt Trung Miên đều không thể tùy tiện	Nhưng lúc này đây, liên kích của hắn lại không thông thuận như ban đầu, bởi vì đây là Miêu Yêu đang cuồng bạo, tốc độ và công kích nhanh đến mức Nguyệt Trung Miên không theo kịp.

7	两人都挺无奈。	Hai người đều rất bất đắc dĩ	Hai người đều thật hết cách rồi.
8	大家齐盯着叶修 的屏幕，就见他的角色 玩命飞奔，时不时 扭头甩一下身后， 于是大家都看到他 身后有大片的追兵， 正是三大公会的玩家 。	Mọi người đồng thời nhìn màn hình của Diệp Tu, chỉ thấy nhân vật có chạy trốn lên, thỉnh thoảng lại nhìn về sau, mọi người đều nhìn thấy hắn có người cướp quái, còn ba công hội lớn	Mọi người đồng thời nhìn chăm chăm màn hình Diệp Tu, chỉ thấy nhân vật liều mạng chạy như bay, chốc chốc lại ngoái cổ nhìn về phía sau, vì thế tất cả mọi người đều thấy được đồng lớn truy bình phía sau hắn, đúng là người chơi của cả ba công hội lớn.



# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết luận

Sau thời gian nghiên cứu, dưới sự hướng dẫn tận tình của TS Nguyễn Văn Vinh, tôi đã hoàn thành luận văn “Nghiên cứu dịch máy Trung – Việt dựa vào mô hình Transformer”. Luận văn đã đạt được các kết quả chính như sau:

- Tìm hiểu và hệ thống các kiến thức liên quan:
  - Tổng quan về dịch máy, một số các cách tiếp cận dịch máy
  - Mạng nơ ron nhân tạo, huấn luyện mạng nơ ron
  - Mô hình dịch máy nơ ron và tập trung nghiên cứu mô hình Transformer
- Cài đặt và thử nghiệm mô hình Transformer và tối ưu các tham số của mô hình và áp dụng cho cặp ngôn ngữ Trung – Việt
- Thử nghiệm so sánh mô hình Transformer với mô hình dịch máy nơ ron sử dụng RNN và Attention
- Trau dồi kiến thức về ngôn ngữ và dịch thuật

## Hướng phát triển tương lai

Với những kiến thức và kỹ năng có được từ khóa luận, trong tương lai tôi sẽ tiếp tục thu thập và bổ sung ngữ liệu cho hệ thống, đồng thời khai thác thêm các đặc trưng ngôn ngữ và thêm các nguồn dữ liệu mở như các từ điển vào hệ thống để nâng cao chất lượng dịch. Đồng thời tìm hiểu thêm các biến thể của mô hình Transformer và thử nghiệm với nhiều cặp ngôn ngữ khác nhau.

# TÀI LIỆU THAM KHẢO

## Tiếng Việt:

- [1] GS. Hồ Tú Bảo, GS. Lương Chi Mai (2005), “Về xử lý tiếng Việt trong công nghệ thông tin”. Available: <http://www.jaist.ac.jp/~bao/Writings/VLSPwhitepaper%20-%20Final.pdf>

## Tiếng Anh:

- [2] A. Vaswani et al., “Attention Is All You Need,” arXiv:1706.03762 [cs], Dec. 2017. Available: <http://arxiv.org/abs/1706.03762>.
- [3] Daniel Jurafsky & James H. Martin, 2006. *Speech and Language Processing: An introduction to natural language processing, computational linguistics, and speech recognition*.
- [4] G. Doddington, “Automatic evaluation of machine translation quality using n-gram co-occurrence statistics,” in Proceedings of the second international conference on Human Language Technology Research, San Diego, California, 2002, p. 138, doi: 10.3115/1289189.1289273.
- [5] Krzysztof Wołk, Krzysztof Marasek, 2015. *Neural-based Machine Translation for Medical Text Domain. Based on European Medicines Agency Leaflet Texts*. ScienceDirect Available at: <http://www.sciencedirect.com/science/article/pii/S1877050915025910>
- [6] M. Popel and O. Bojar, “Training Tips for the Transformer Model,” The Prague Bulletin of Mathematical Linguistics, vol. 110, no. 1, pp. 43–70, Apr. 2018
- [7] M.-T. Luong, H. Pham, and C. D. Manning, “Effective Approaches to Attention-based Neural Machine Translation,” arXiv:1508.04025 [cs], Sep. 2015. Available: <http://arxiv.org/abs/1508.04025>.
- [8] M.-T. Luong, Q. V. Le, I. Sutskever, O. Vinyals, and L. Kaiser, “Multi-task Sequence to Sequence Learning,” arXiv:1511.06114 [cs, stat], Mar. 2016. Available: <http://arxiv.org/abs/1511.06114>.
- [9] P.-C. Chang, M. Galley, and C. D. Manning, “Optimizing Chinese word segmentation for machine translation performance,” in Proceedings of the Third Workshop on Statistical Machine Translation - StatMT '08, Columbus, Ohio, 2008, pp. 224–232.
- [10] P. Koehn, “Neural Machine Translation,” arXiv:1709.07809 [cs], Sep. 2017. Available: <http://arxiv.org/abs/1709.07809>.
- [11] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, Alexander M. Rush. 2017. OpenNMT: Open-Source for Neural Machine Translation. Proceedings of AMTA 2018, vol. 1: MT Research Track.

- [12] T. Vu, D. Q. Nguyen, D. Q. Nguyen, M. Dras, and M. Johnson, “VnCoreNLP: A Vietnamese Natural Language Processing Toolkit,” in Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations, New Orleans, Louisiana, Jun. 2018, pp. 56–60, doi: 10.18653/v1/N18-5012.
- [13] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient Estimation of Word Representations in Vector Space,” arXiv:1301.3781 [cs], Sep. 2013. Available: <http://arxiv.org/abs/1301.3781>.
- [14] R. Sennrich and B. Zhang, “Revisiting Low-Resource Neural Machine Translation: A Case Study,” in Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, Jul. 2019, pp. 211–221, doi: 10.18653/v1/P19-1021.