TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

Ngô Phù Hữu Đại Sơn

Sử dụng GNN để tăng độ hiệu quả của bài toán Machine Translation

KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN
CHƯƠNG TRÌNH CHÍNH QUY

Tp. Hồ Chí Minh, tháng 07/2022

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

Ngô Phù Hữu Đại Sơn - 18120078

Sử dụng GNN để tăng độ hiệu quả của bài toán Machine Translation

KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN
CHƯƠNG TRÌNH CHÍNH QUY

NGƯỜI HƯỚNG DẪN

TS. Nguyễn Ngọc Thảo

ThS. Tạ Việt Phương

Tp. Hồ Chí Minh, tháng 07/2022

Lời cảm ơn

Trải qua thời gian dài học tập trong trường, đã đến lúc những kiến thức của em được vận dụng vào thực tiễn công việc. Em lựa chọn làm khóa luận tốt nghiệp để tổng hợp lại kiến thức của mình. Đề tài của em là: "*Sử dụng GNN để tăng độ hiệu quả của bài toán Machine Translation*". Trong suốt quá trình làm khóa luận, em đã nhận được sự hướng dẫn, giúp đỡ quý báu của các thầy cô, các anh chị và các bạn. Em xin được bày tỏ lời cảm ơn chân thành tới:

TS Nguyễn Ngọc Thảo đã hướng dẫn và truyền đạt những kinh nghiệm quý báu cho em trong suốt thời gian làm khóa luận tốt nghiệp của mình.

Em cũng cảm ơn gia đình và bạn bè đã giúp đỡ em hoàn thành tốt khóa luận.

Khóa luận của em còn những hạn chế về năng lực và những thiếu sót trong quá trình nghiên cứu. Em xin lắng nghe và tiếp thu những ý kiến của giáo viên phản biện để hoàn thiện, bổ sung kiến thức.

Em xin chân thành cảm ơn!

Đề cương chi tiết

Thông tin chung

- **Người hướng dẫn:**
  - TS. Nguyễn Ngọc Thảo (Khoa Công nghệ thông tin)
  - ThS. Tạ Việt Phương (Trường Đại học Công nghệ Thông tin)
- **Sinh viên thực hiện:**
  1. Ngô Phù Hữu Đại Sơn (MSSV: 18120078)
- **Loại đề tài:** Nghiên cứu
- **Thời gian thực hiện:** Từ *01/2022* đến *07/2022*

Nội dung thực hiện

Giới thiệu về đề tài

Các mô hình giải quyết bài toán Machine Translation sử dụng kiến trúc Attention đang thể hiện các kết quả rất tốt trong các thí nghiệm và thực tế. Một trong những phần quan trọng của những mô hình này là các mô hình encoding các từ thành các word embeddings để có thể chuyển từ ngữ thành các đầu vào có thể tính toán.

Đồ thị xuất hiện một cách tự nhiên trong nhiều lĩnh vực ứng dụng, từ phân tích xã hội, sinh học, hóa học đến thị giác máy tính. Đồ thị cho phép nắm bắt các mối quan hệ cấu trúc giữa các dữ liệu và do đó cho phép thu thập nhiều thông tin chi tiết hơn.

Một trong các phương pháp giúp encode các từ ngữ thành các vector là sử dụng Graph Convolution Neural NetWork (GCN). Lợi ích của việc sử dụng *GCN* là:

- Thể hiện được các Syntactic Context giữa các từ ngữ trong câu.
- Thể hiện được các Semantic Context giữa các từ ngữ với nhau.

Nhờ vào đó, các embeddings được học ra có khả năng biểu diễn ngữ nghĩa và cấu trúc trong câu tốt hơn, phù hợp cho các bài toán về Machine Translation.

Mục tiêu đề tài

Mục tiêu của đề tài này bao gồm: (1) - Nghiên cứu, khỏa sát các thuật toán biểu diễn từ ngữ thành các embeddings và (2) - Kết hợp cài đặt các thuật toán giúp nâng cao hiệu xuất của mô hình *transformer* để giải quyết bài toán Machine Translation.

Việc nghiên cứu khảo sát các thuật toán nhằm đưa ra được các đánh giá về ưu điểm và nhược điểm của chúng. Từ đó, cho thấy được độ hiệu quả và tiềm năng của *GCN* trong bài toán machine translation.

Việc kết hợp cài đặt nhằm chứng minh tính hiệu quả của mô hình đề xuất.

Phạm vi của đề tài

Nghiên cứu ở ([11]) đã chỉ các các hướng nghiên cứu cho mô hình transfomer. Trong đó: (1) - Cải thiện hiệu suất cho mô hình *transformer* ,

(2) - Khái quát hóa mô hình giúp huấn luyện với tập dữ liệu nhỏ hơn. (3) - Tăng độ thích nghi của mô hình vào các tác vụ thực tế hơn.

Đề tài của khóa luận này có phạm vi nghiên cứu giúp cải thiện hiệu suất của mô hình *transformer* dựa trên embeddings đã được huấn luyện trước([14])

Đề tài thực hiện bài toán dịch cụ thể từ tiếng Anh sang tiếng Đức.

Cách tiếp cận dự kiến

Phương pháp chính. Theo các đề xuất ở ([14]):

- Mô hình *SynGCN* để huấn luyện các word embeddings theo Syntactic Context.
- Sau đó sử dụng mô hình *SemGCN* để huấn luyện các word embeddings từ *SynGCN* theo Semantic Context.

Phương pháp đề xuất trong khóa luận nhắm tích hợp các embeddings sau khi được vào huấn luyện ở mô hình *SynGCN+SemGCN* sẽ được sử dụng để huấn luyện mô hình *transformer* được đề xuất ở ([15]) với kì vọng sẽ tăng được hiệu xuất của mô hình.

Dữ liệu thực nghiệm. Sử dụng tập dữ liệu Multi30k để huấn luyện mô hình có thể dịch từ tiếng Anh sang tiếng Đức. Sử dụng 80% dữ liệu để huấn luyện và 20% dữ liệu để kiểm thử.

Phương pháp đối sánh. Sử dụng mô hình transfomer làm mô hình baseline. So sánh hiệu suất giữa mô hình đề xuất và mô hình baseline.

Kết quả dự kiến của đề tài

Sau khi tiến hành, nghiên cứu này kỳ vọng sẽ đạt được các kết quả sau:

- Nắm được ý tưởng của việc sử dụng *GCN* để huấn luyện các word embeddings.

- Cài đặt được mô hình baseline.
- Tích hợp *SynGCN* và *SemGCN* vào mô hình *transformer* . So sánh hiệu suất của mô hình đề xuất với mô hình baseline.

Kế hoạch thực hiện

Kế hoạch thực hiện khóa luận bao gồm các giai đoạn được trình bày như sau:

Giai đoạn	**Thời gian**	**Công việc**
1	01/01/2022 - 31/01/2022	Tìm hiểu kiến thức nền tảng về transformer Cài đặt mô hình transformer
2	01/02/2022 - 28/02/2022	Tìm hiểu kiến thức nền tảng về word embeddings Tìm hiểu kiến thức nền tảng về GCN Tìm hiểu kiến thức nền tảng của SynGCN và SemGCN
3	01/03/2022 - 31/03/2022	Tích hợp mô hình SynGCN và SemGCN vào mô hình transformer
5	01/04/2022 - 30/04/2022	Chạy các thực nghiệm trên mô hình Phân tích và đánh giá kết quả
6	01/05/2022 - 31/05/2022	Viết luận văn Làm slide thuyết trình Tập thuyết trình

Bảng 1: Bảng kế hoạch thực hiện

Mục lục

Lời cảm ơn **i**

Đề cương chi tiết **ii**

Mục lục **vi**

Tóm tắt **x**

1 Giới thiệu **1**

1.1 Đặt vấn đề 1

1.2 Bài toán dịch máy (Machine Translation) 2

1.3 Các cách tiếp cận trước 3

1.4 transformer 5

1.5 Tokenization & Word Embeddings 7

1.6 Graph Convolution Network (GCN) 8

1.7 WordGCN 10

2 Tổng quan lý thuyết **13**

2.1 Mô hình transformer dịch máy 13

2.1.1 Tổng quan mô hình 13

2.1.2 Lớp mã hóa (Encoder) 14

2.1.3 Lớp giải mã 21

2.1.4 residuals 25

2.1.5 Tổng kết mô hình 26

2.2 Mô hình WordGCN huấn luyện word embeddings cho kho ngữ liệu 27
2.2.1 Lý thuyết đồ thị cơ bản 27
2.2.2 Mô hình Graph Convolution Network và bài toán representation learning 32
2.2.3 Mô hình WordGCN 34

3 Phương pháp đề xuất **38**
3.1 Áp dụng mô hình WordGCN vào mô hình Transformer . . 38
3.2 Chuẩn bị dữ liệu 39
3.2.1 Tập dữ liệu WMT14 39
3.2.2 Tiền xử lý dữ liệu cho mô hình SynGCN 41
3.2.3 Tiền xử lý dữ liệu cho mô hình SemGCN 43
3.2.4 Tiền xử lý dữ liệu cho mô hình transformer 45

4 Huấn luyện và Kết quả **48**
4.1 Huấn luyện 48
4.1.1 Kích thước batch của dữ liệu huấn luyện 48
4.1.2 Kết quả 49

5 Kết luận **50**

Tài liệu tham khảo **52**

Danh sách hình

1.1 Phân bố các ngôn ngữ trên thế giới[9] 2
1.2 Tổng quan kiến trúc mô hình Seq2Seq [4] 3
1.3 Kiến trúc *mạng hồi quy* với các thông tin ẩn 4
1.4 Kiến trúc *LSTM* [8] 5
1.5 Tổng quan kiến trúc của *Transformer* 6
1.6 Tokenization & word embeddings 7
1.7 Các phân tử hóa học được biểu diễn dưới dạng đồ thị . . . 8
1.8 So sánh *CNN* và *GCN* 10
1.9 Thông tin cú pháp của một câu được biểu diễn dưới dạng đồ thị 11
1.10 Thông tin ngữ nghĩa của các từ được biểu diễn dưới dạng đồ thị 12

2.1 Minh họa kiến trúc Encoder-Eecoder [5] 13
2.2 *Bộ mã hóa* và *bộ giải mã* trong *Transformer*[5] 14
2.3 Kiến trúc lớp mã hóa[5] 15
2.4 Minh họa cơ chế *Self-attention*[5] 16
2.5 Đồ thị hàm *softmax* và đạo hàm của một thành phần trong tập phân loại 17
2.6 Minh họa cơ chế *Multihead attention* 18
2.7 Cơ chế tổng hợp feed-forward trong Multihead attention[5] 19
2.8 Minh họa kiến trúc của *Multihead attention*[16] 20
2.9 Đồ thị positional embedding [13] 21
2.10 Minh họa kiến trúc bộ lớp mã hóa [5] 22

2.11 So sánh *self attention* và *maked self attention* 22
2.12 Minh họa cơ chế *encoder-decoder-atention* [5] 24
2.13 Minh họa kiến trúc *residual*[5] 26
2.14 Tổng kết mô hình *Transformer* [5] 27
2.15 Phân biệt đơn đồ thị, đa đồ thị và giả đồ thị 28
2.16 Phân biệt đồ thị vô hướng và đồ thị có hướng 29
2.17 Phân biệt đồ thị có trọng số và không trọng số 29
2.18 Minh họa về vector đặc trưng 30
2.19 Ma trận kề 31
2.20 Minh họa về bài toán *Representation learning* 32
2.21 Mô hình cơ bản của *SynGCN* [14] 35
2.22 Ý tưởng cơ bản của *SemGCN* [14] 36

3.1 Minh họa lớp mã hóa embedding với số lượng từ vựng trong kho ngữ liệu $n_{voc} = 3$ và số chiều của embedding $d = 2$ . . 39
3.2 Ảnh minh họa thượng vị và hạ vị 44
3.3 Minh họa về các token đặc biệt trong mô hình *transformer* 46
3.4 Định dạng của tập tin chứa các *embedding* được huấn luyện bởi mô hình *WordGCN*. Ở ví dụ này mỗi từ được biểu diễn bởi một embedding của số chiều là 20. 46

Danh sách bảng

1 Bảng kế hoạch thực hiện v

3.1 Bảng thống kê số liệu của kho ngữ liệu song ngữ *Europarl* 40
3.2 Bảng thống kê số liệu của kho ngữ liệu song ngữ *New Commentary* 40
3.3 40
3.4 Bảng thống kê số liệu của kho ngữ liệu song ngữ *Common Crawl* 41

4.1 Bảng kết quả huấn luyện của mô hình cơ sở và mô hình đề xuất 49

Tóm tắt

Trong quá trình phát triển của công nghệ ngày nay, nhu cầu được kết nối với mọi người, mọi loại thông tin trên khắp thế giới là một nhu cần thiết cho tất cả mọi người. Một trong những rào cản để kết nối giữa các thực thể ở cách xa nhau về vùng địa lý chính là ngôn ngữ. Do đó, bài toán về dịch máy ngày càng trở nên quan trọng và cần được xử lý một cách hiệu quả.

Mô hình *Transformer* là một trong các mô hình được đánh giá là cho ra kết quả tốt cho bài toán dịch máy vào những năm gần đây. Mô hình hoàn toàn dựa trên cơ chế attention thay vì sử dụng các kiến trúc hồi quy hay tích chập vốn đã cho ra các kết quả tốt trước đây.

Trong những năm gần đây, các thuật toán học máy đang được phát triển trên các bộ dữ liệu đồ thị. Trong đó, có kiến trúc mạng *Graph Convolutional Network (GCN)* được đánh giá là một trong những kiến trúc quan trọng trong việc giải các bài toán học máy trên đồ thị.

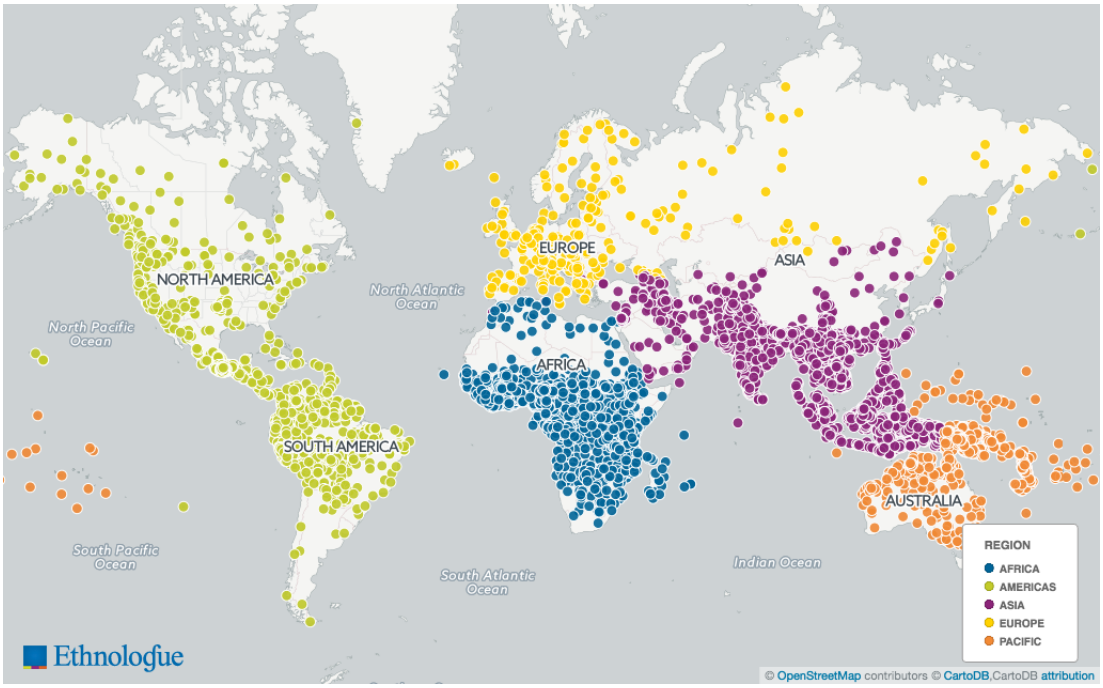
Hướng tiếp cận của khóa luận này là áp dụng được mô hình *Graph Convolutional Network* vào mô hình dịch máy (cụ thể là *Transformer*) để tăng tính hiệu quả và tăng tốc độ hội tụ của mô hình.

Chương 1

Giới thiệu

1.1 Đặt vấn đề

Ngôn ngữ là một loại phương tiện giúp con người có thể giao tiếp và truyền đạt suy nghĩ, ý kiến của mình cho những người xung quanh. Theo trang *Ethnologue.com*[9], tính đến năm 2022, trên thế giới có 7151 ngôn ngữ. Vì vậy, một người không thể nào học và hiểu hết mọi ngôn ngữ trên thế giới. Từ đó, có thế độ cần thiết của việc dịch từ một ngôn ngữ sang một ngôn ngữ khác. Ngành khoa học máy tính, cụ thể hơn là xử lý ngôn ngữ tự nhiên, chúng ta cũng quan tâm đến bài toán trên.



Hình 1.1: Phân bố các ngôn ngữ trên thế giới[9]

1.2 Bài toán dịch máy (Machine Translation)

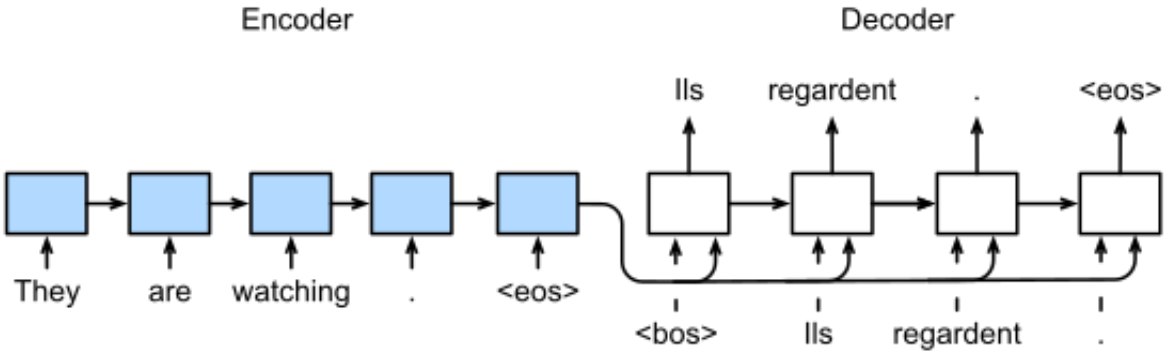
Bài toán dịch máy là một lĩnh vực trong ngành khoa học máy tính. Đầu vào được nhập vào máy tính là một đoạn văn bản từ một ngôn ngữ (ngôn ngữ nguồn). Và qua quá trình xử lý, đưa ra được đoạn văn bản tương ứng ở một ngôn ngữ khác (ngôn ngữ đích). Khác với các mô hình xử lý ngôn ngữ khác, khi chúng chỉ cần một kho ngữ liệu của một ngôn ngữ. Các mô hình dịch máy cần ít nhất hai kho ngữ liệu của ngôn ngữ nguồn và ngôn ngữ đích. kho ngữ liệu bao gồm tập các đoạn văn bản. Với mỗi đoạn văn bản từ ngôn ngữ nguồn sẽ được ánh xạ đến một đoạn văn bản có ý nghĩa tương ứng ở ngôn ngữ đích. Các kho ngữ liệu này có thể tổng hợp từ các nguồn khác nhau: từ các bản dịch (subtitle) của các bộ phim, bản dịch sách và cả các bộ dữ liệu được thiết kế riêng cho bài toàn dịch máy (các bản dịch từ các chuyên gia).

Để có thể hiểu hơn sâu hơn bài toán và tìm ra hướng giải quyết, ta cần phải hiểu được cách tự nhiên mà con người dịch một đoạn văn bản từ ngôn ngữ nguồn sang ngôn ngữ đích như thế nào. Theo như [6], quá trình này có thể chia làm hai bước:

- Trích xuất ngữ cảnh, văn cảnh (*context*) của ngôn ngữ nguồn thành thông tin.
- Chuyển hóa thông tin thu thập được thành ngôn ngữ đích.

1.3 Các cách tiếp cận trước

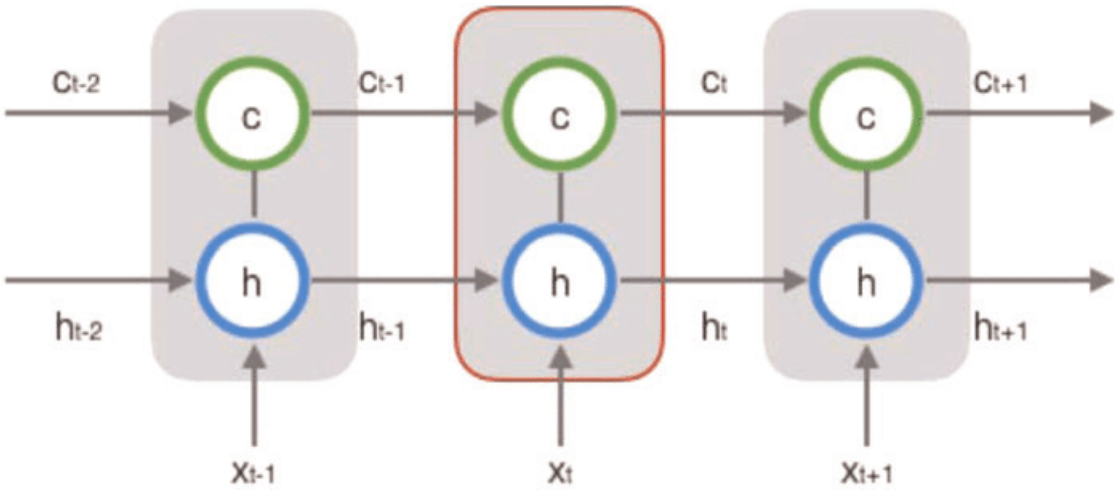
Với tính chất trên của bài toán, ta có thể thấy bài toán này là một bài toán *sequence-to-sequence(Seq2Seq)* và có thể giải quyết bằng kiến trúc *encoder-decoder*. Các mô hình sử dụng các mạng nơ ron hồi quy (Recursive Neural Network) sử dụng cơ chế Long-Short-Term-Memory.



Hình 1.2: Tổng quan kiến trúc mô hình Seq2Seq [4]

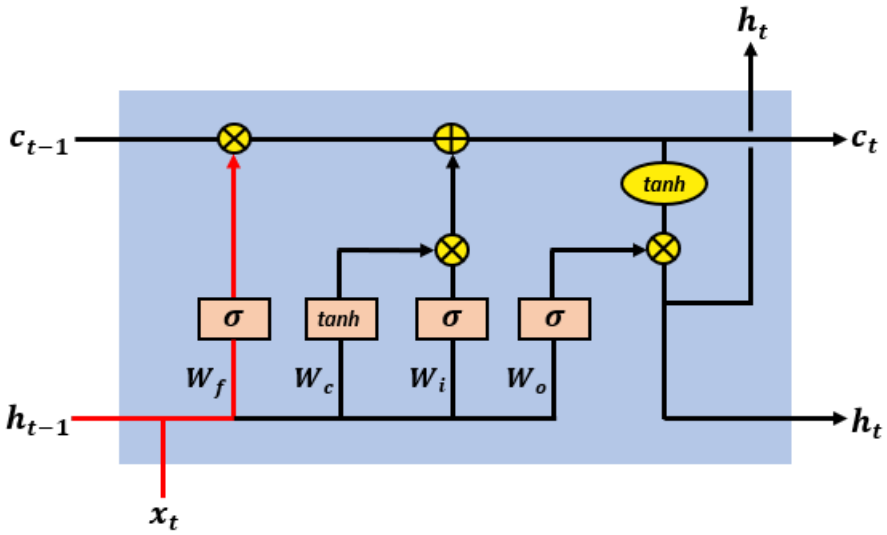
Các mô hình hồi quy (*Recurrent model*) cho các kết quả tốt trong bài toán dịch máy. Tuy nhiên, các mô hình này lại sử dụng cơ chế hồi quy. Ở mỗi bước tính toán, mô hình sử dụng thông tin ẩn (*hidden state*) được tổng hợp từ đầu văn bản đến hiện tại h_t để làm đầu vào tính toán. Quá trình này lặp lại cho mỗi bước. Từ đó, ta có thể thấy mô hình tính toán bước tiếp theo phải phụ thuộc vào bước trước đó, dẫn đến không thể song

song quá trình tính toán này được. Điều này khiến cho việc tối ưu thời gian huấn luyện lẫn hiệu quả tính toán của mô hình trước nên khó khăn.



Hình 1.3: Kiến trúc *mạng hồi quy* với các thông tin ẩn

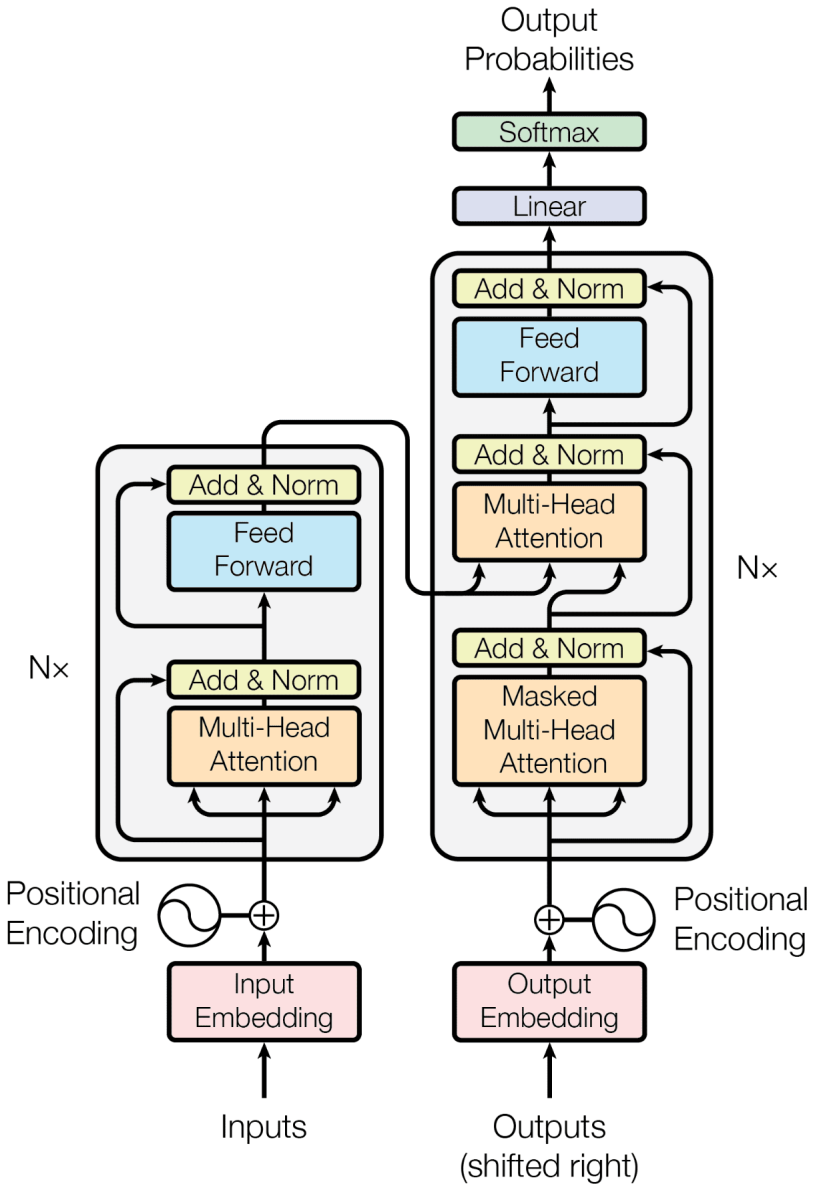
Một nhược điểm khác của các mô hình hồi quy đó là dễ xảy ra hiện tượng *Gradient biến mất* (*Gradient Vanishing*) và đôi khi là *Gradient bùng nổ* (*Gradient Exploding*). Nguyên nhân cốt lõi của vấn đề này là do công thức của hàm phi tuyến chưa hợp lý làm cho quá trình lan truyền ngược (*back propagation*), các giá trị *gradient* của mỗi lớp quá lớn hoặc quá nhỏ. Do đó làm cho *gradient* lan truyền về sau trở nên càng lớn hoặc càng nhỏ theo cấp số mũ. Vấn đề *Gradient biến mất* sẽ làm cho mô hình hồi tụ chậm đi khi mô hình học sâu được tăng thêm số lớp. Để giải quyết vấn đề này, kiến trúc *LSTM (long short term memory)* được giới thiệu. Với *LSTM*, hiện tượng *Gradient biến mất* được cải thiện. Do đó khi mô hình đang xử lý ở bước thứ i, các thông tin từ các bước trước đó không bị mất đi. Nhờ vậy mà quá trình học ở bước hiện tại trở nên hiệu quả hơn.



Hình 1.4: Kiến trúc *LSTM* [8]

1.4 transformer

Transformer [15] - một phương pháp dựa hoàn toàn trên cơ chế attention, bỏ qua các cấu trúc của mạng nơ ron hồi quy và mạng nơ ron tích chập phức tạp, giúp đơn giản hóa mô hình những vẫn thể hiện được độ hiệu quả của mô hình. Theo **Paper attention is all your need** [15], mô hình cho kết quả 41.8 BLEU khi huấn luyện trên tập WMT 2014, cao hơn tất cả các mô hình dịch máy trước đó.



Hình 1.5: Tổng quan kiến trúc của *Transformer*

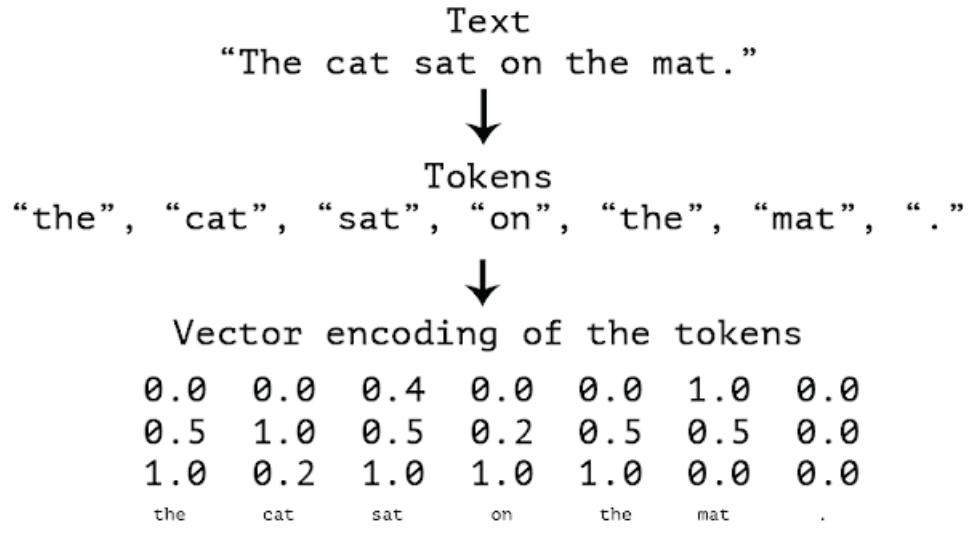
Nhờ không sử dụng kiến trúc hồi quy, *Transformer* tránh được nhược điểm chí mạng của các mô hình loại này. Dựa hoàn toàn vào cơ chế *attention*, cụ thể hơn là giới thiệu kiến trúc *Hultihead attention* giúp việc song song tính toán mô hình. Từ đó mà tăng độ hiệu quả huấn luyện cũng như hiệu quả tính toán.

1.5 Tokenization & Word Embeddings

Trong bài toán dịch máy, các đoạn văn bản thô cần phải được tiền xử lý, chuyển đổi thành các dạng dữ liệu mà máy tính có thể hiểu được. Quá trình đó được thực hiện như sau:

- *Tokenization* là quá trình tách đoạn văn bản ra thành các từ thành phần (*token*). Các token này đã được quy định sẵn trong một tập các từ đã biết trước được gọi là kho từ vựng (*vocabulary*). Với các từ lạ, không thuộc trong kho từ vựng sẽ được đánh dấu là *<unk> (unknown).*

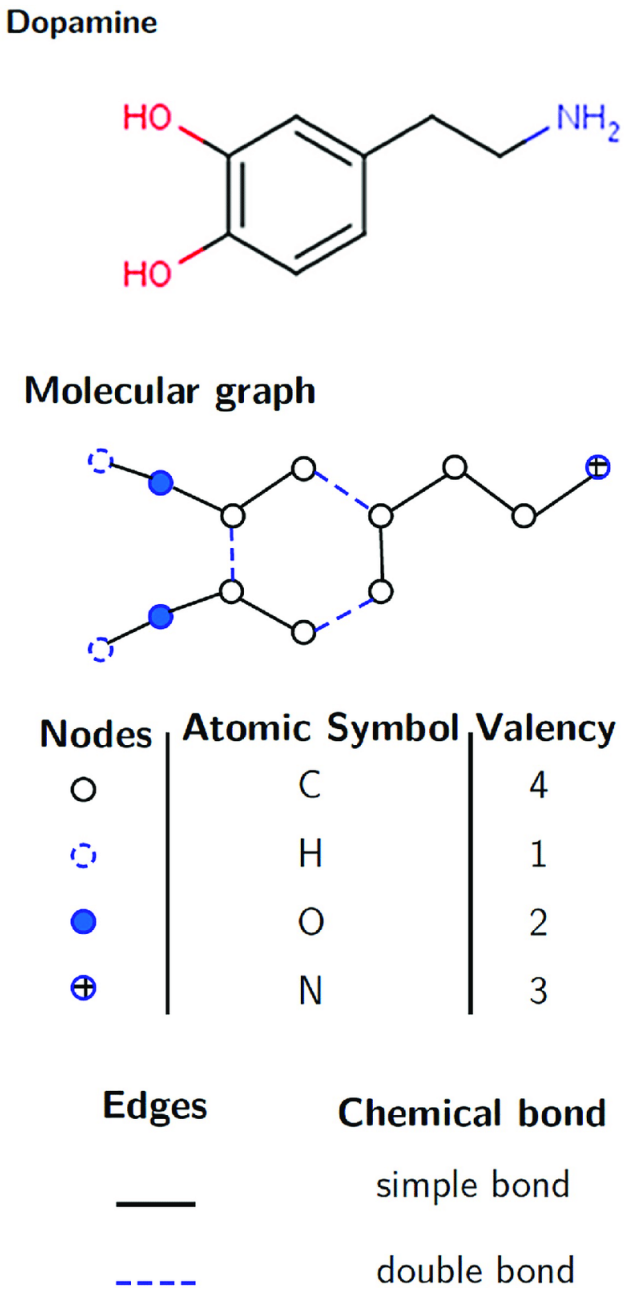
- Các *token* sau khi được tách ra vẫn chưa thể đưa vào mô hình do chúng vẫn ở dạng dữ liệu mà các mô hình chưa thể hiểu. Do đó, với mỗi *token*, ta cần chuyển chúng thành các vector N chiều (N chọn trước) mang tính chất và đại diện cho từ đó. Với mỗi chiều của vector được biểu diễn bằng một số thực. Các vector này được gọi là các *Word Embeddings*. Các *Word Embeddings* sẽ là đầu vào của mô hình. Một số phương pháp để tính toán các *Word Embeddings* trước đây gồm: *CBOW, N-gram, skip-gram,...*



Hình 1.6: Tokenization & word embeddings

1.6 Graph Convolution Network (GCN)

Khai thác quan hệ giữa các điểm dữ liệu với nhau là một đề tài được nhiều sự quan tâm trong học máy. Trong các mô hình học sâu trước đây, ta chỉ có thể trích xuất được các mối quan hệ thông thường từ các bộ dữ liệu *Euclidian*.



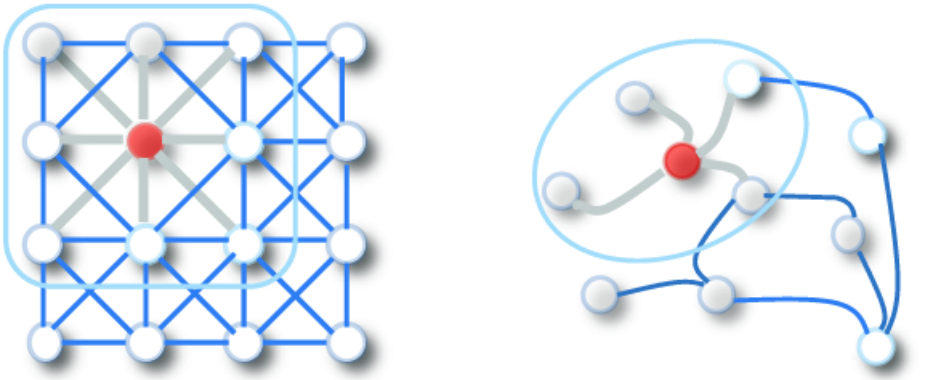
Hình 1.7: Các phân tử hóa học được biểu diễn dưới dạng đồ thị

Trong thực tế, không phải mọi dữ liệu đều biểu diễn ở dạng *Euclidian*. Do đó, để khai thác được thông tin từ các bộ dữ liệu đồ thị (*non-Euclidian*), ta cần sử dụng các mô hình *Graph Neural Network (GNN)*. Trong vài năm qua, nhiều biến thể của GNN đã được phát triển. Trong đó, *Graph Convolution Network (GCN)* là một biến thể quan trọng được xem như là biến thể cơ bản nhất của *Graph Neural Network(GNN)*.

Graph Convolutional Network ứng dụng phép tích chập được giới thiệu trong *convolution layer* của *mạng tích chập(CNN)*:

- Đối với phép tích chập trên *CNN*, một đoạn dữ liệu của lớp hiện tại sẽ được nhân vô hướng (tích chập) với một bộ lọc (*filter* hoặc *kernel*). Bộ lọc sẽ trượt trên bộ dữ liệu và các đầu ra của phép tích chập sẽ được truyền vào lớp tiếp theo của mạng.

- Với phép tích chập trên *GCN*. Ta xét từng nút(*node*) của đồ thị. Với mỗi nút, ta nhóm nút này với các nút kề của chính nó lại và thực hiền phép tích chập tương tự với một màn lọc. Do số lượng nút kề của mỗi nút là khác nhau, nên kích thước của *filter* sẽ không cố định. Phép tích chập sẽ được thực hiện trên tất cả các nút của đồ thị.



(a) 2D Convolution. Analogous to a graph, each pixel in an image is taken as a node where neighbors are determined by the filter size. The 2D convolution takes the weighted average of pixel values of the red node along with its neighbors. The neighbors of a node are ordered and have a fixed size.

(b) Graph Convolution. To get a hidden representation of the red node, one simple solution of the graph convolutional operation is to take the average value of the node features of the red node along with its neighbors. Different from image data, the neighbors of a node are unordered and variable in size.

Fig. 1: 2D Convolution vs. Graph Convolution.

Hình 1.8: So sánh *CNN* và *GCN*

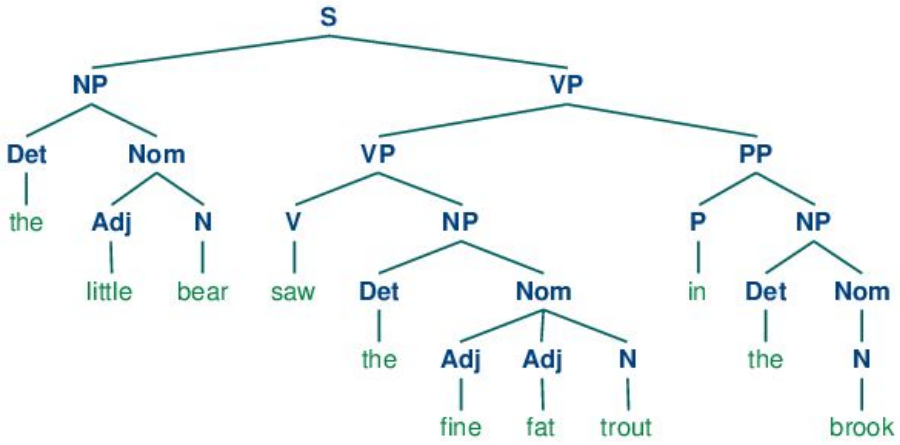
Nhận xét về phép tích chập 2 chiều của mô hình *CNN*. Nếu ta xem mỗi điểm dữ liệu là một đỉnh của đồ thị và các điểm dữ liệu xung quanh sẽ có cạnh nối đến với đỉnh tương ứng. Phép tích chập của *CNN* có ý nghĩa hoàn toàn giống với phép tích chập của mô hình GCN.

1.7 WordGCN

Quan hệ của các từ trong một ngôn ngữ là dạng quan hệ non-Euclidian. Do đó, trích xuất thông tin của chúng bằng các mô hình *GCN* được kì vọng là có kết quả tốt hơn các phương pháp trước đó. Các loại thông tin có thể khai thác bao gồm:

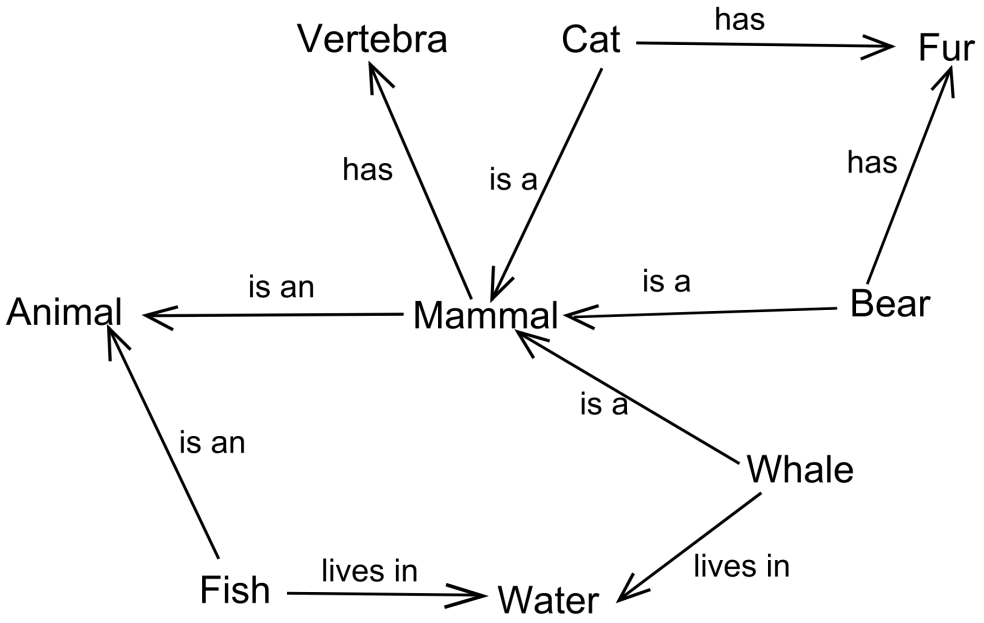
- Thông tin về cú pháp (*syntactic context*) của ngôn ngữ. Các từ trong một câu sẽ có các quy định về các mối quan hệ của chúng trong câu.

Biểu diễn các mối quan hệ này là các cạnh còn các từ trong câu là các nút của đồ thị. Từ đó mà ta có thể khai thác được các đặc trưng cú pháp của các từ. Mô hình *SynGCN* sẽ giúp ta huấn luyện được các *embeddings* mang các thông tin trên.



Hình 1.9: Thông tin cú pháp của một câu được biểu diễn dưới dạng đồ thị

- Thông tin về ngữ nghĩa (*semantic context*) của ngôn ngữ. Ý nghĩa của các từ sẽ có các quan hệ với nhau như: đồng nghĩa, trái nghĩa, kế thừa,... Nhờ các mối quan hệ đó, ta có thể biểu diễn được đồ thị ngữ nghĩa của các từ trong bộ từ vựng. Từ đó mà ta có thể khai thác được các đặc trưng ngữ nghĩa của các từ. Mô hình *SemGCN* sẽ giúp ta huấn luyện được các *embeddings* mang thông tin ngữ nghĩa.



Hình 1.10: Thông tin ngữ nghĩa của các từ được biểu diễn dưới dạng đồ thị

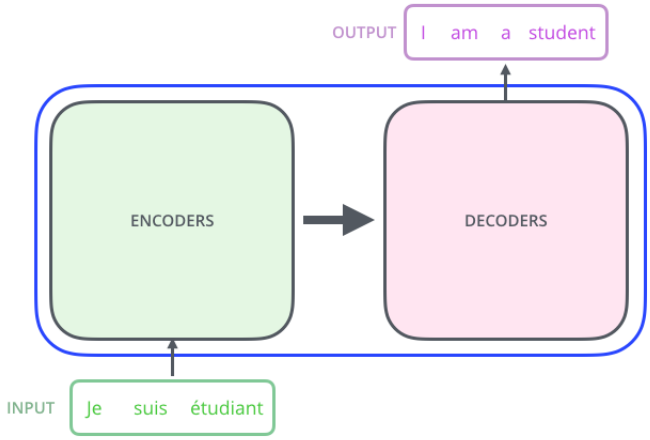
Chương 2

Tổng quan lý thuyết

2.1 Mô hình transformer dịch máy

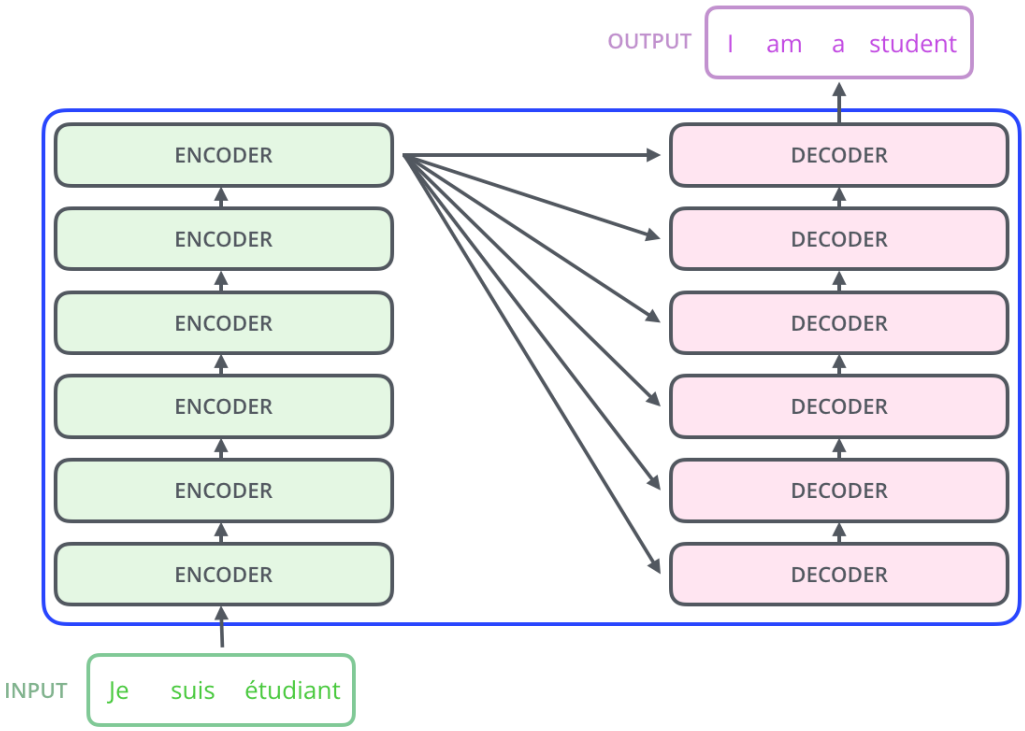
2.1.1 Tổng quan mô hình

Transformer là một mô hình có kiến trúc *Encoder-Decoder*. Mô hình bao gồm 2 thành phần chính là *Encoder (bộ mã hóa)* và *Decoder(bộ giải mã)*. Khi đưa một đoạn văn bản nguồn vào, mô hình sẽ xử lý vào đưa ra đoạn văn bản có ngữ nghĩa tương ứng ở ngôn ngữ đích.



Hình 2.1: Minh họa kiến trúc Encoder-Eecoder [5]

Cụ thể hơn, Bộ mã hóa sẽ bao gồm nhiều lớp mã hóa xếp chồng lên nhau. Theo [15], họ sử dụng 6 lớp mã hóa để tạo thành một bộ mã hóa. Bộ giải mã, tương tự cũng được xếp chồng bởi các lớp giải mã. Số lượng lớp của 2 thành phần phải bằng nhau.



Hình 2.2: *Bộ mã hóa* và *bộ giải mã* trong *Transformer*[5]

2.1.2 Lớp mã hóa (Encoder)

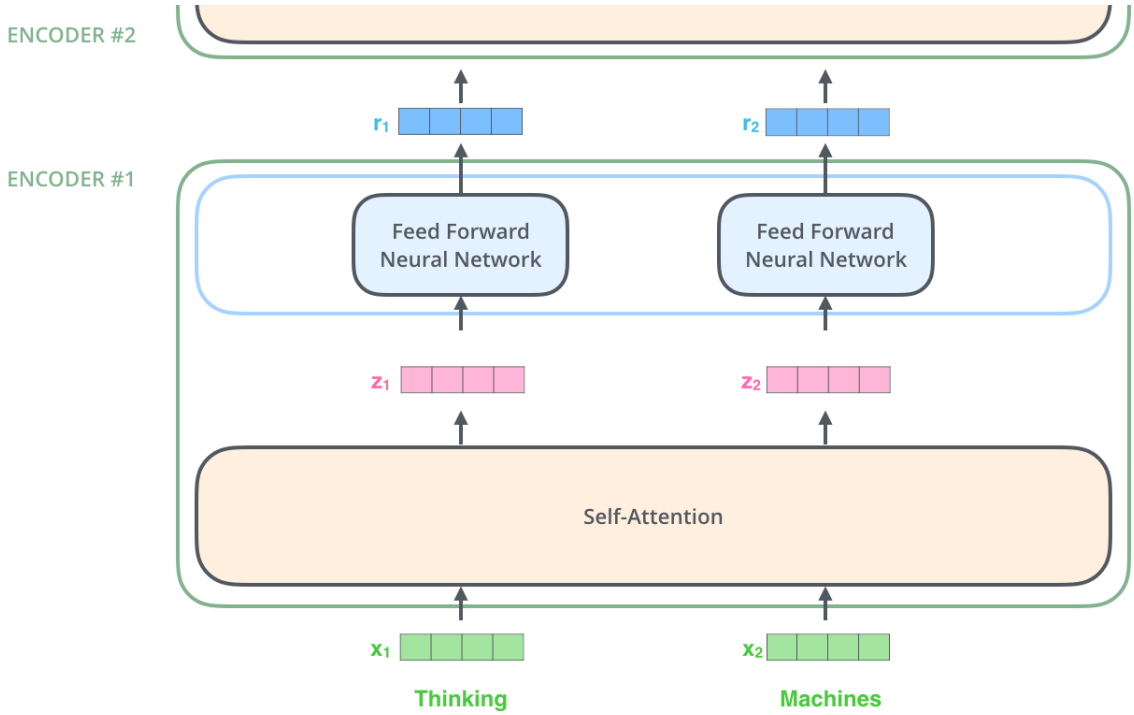
Các lớp của bộ mã hóa là độc lập nhau và có cấu trúc tương tự nhau. Đầu vào của lớp đầu tiên sẽ là các *word embeddings* đã được xử lý qua lớp *positional encoding*, các lớp phía trên sẽ có đầu vào là các vector đầu ra từ lớp ngay dưới. Các vector đầu vào của các lớp này được gọi là *context vector*.

Với mỗi lớp mã hóa sẽ bao gồm 2 lớp con:

- *Self-attetion*: Với mỗi từ trong đoạn văn bản, xem xét độ liên quan của nó với các từ khác trong câu đầu vào. Đưa ra phân bố xác xuất

đối với từng từ một.

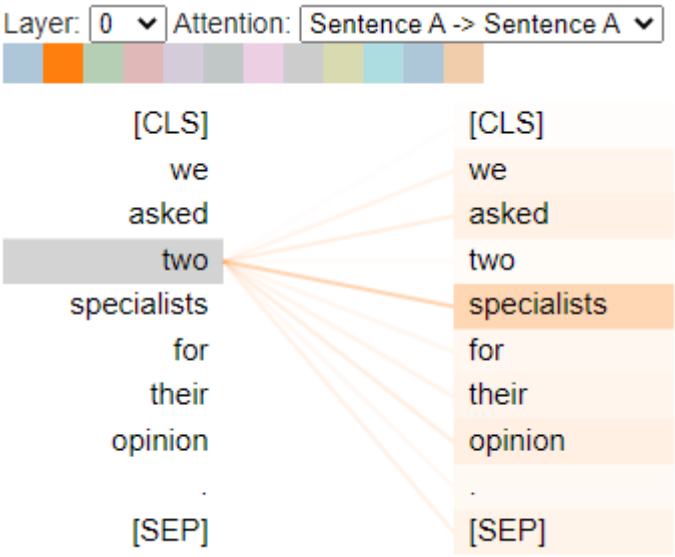
- *Feed forwarding*: Mã hóa phân bố xác xuất tính toán được thành các context vector để đưa vào các lớp mã hóa phía trên.



Hình 2.3: Kiến trúc lớp mã hóa[5]

Cơ chế Self-attention

Self-attention là một cơ chế mới được giới thiệu trong **attention is all need** [15]. Cơ chế này khác với cơ chế *attention* trong các mạng *Seq2Seq* trước đó. *Self-attention* cho phép ta biểu diễn lại mối quan hệ giữa các từ trong một đoạn văn bản.



Hình 2.4: Minh họa cơ chế *Self-attention*[5]

Hình trên cho ta một minh họa về cơ chế *self-attention*. Xét từ *"two"*, theo tư duy của con người, ta sẽ phân tích xem từ "two" trong câu đang có quan hệ gì với những từ khác. Ngoài ra, ta còn xem xét tầm quan trọng của các từ khác tác động lên ý nghĩa của câu. Từ đó mà ta có thể có được một cái nhìn rõ ràng hơn về vai trò của từ này trong câu.

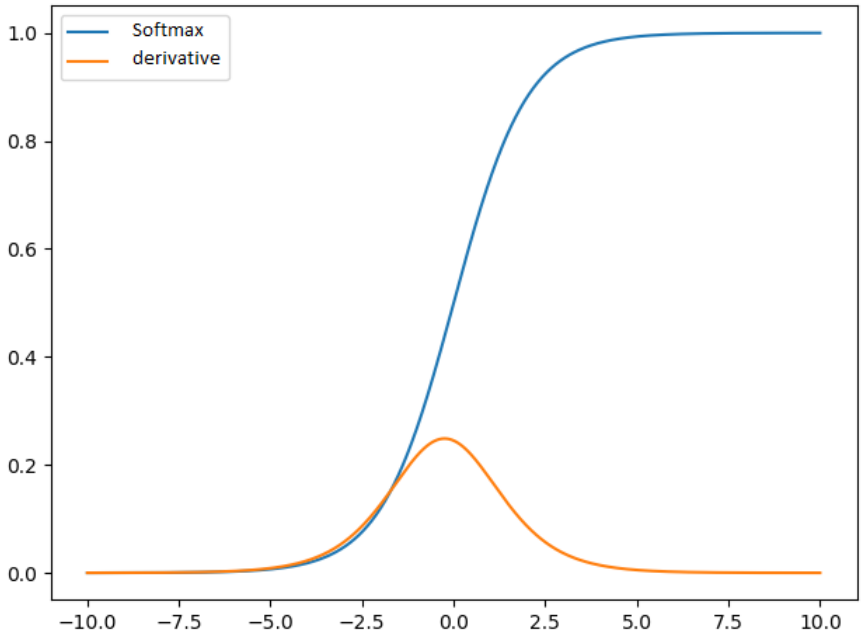
Cụ thể hơn, khi mô hình xử lý từ *"two"*, *self-attention* cho ta thấy được mối liên kết của nó với từ *"specialists"*. Thể hiện vai trò của nó là dùng để chỉ số lượng của các chuyên gia là hai.

Việc tính toán ở mỗi lớp *Self-attention* được tính toán dựa trên công thức sau:

$$Attention = Softmax(\frac{QK^T}{\sqrt{d}})V$$

Trong đó, các ma trận Q, K, V lần lượt là các ma trận *Query, Key, Value*. Hai ma trận Q và K được dùng để tính toán mối quan hệ giữa các từ trong câu. Còn mỗi dòng thứ của ma trận V đại diện cho từ thứ i trong câu. Từ phân bố *softmax*, ta tính được *context vector*, tổng hợp được các thông tin của từ hiện tại và các từ liên quan đến nó.

Trong công thức, ta thấy được tích vô hướng của ma trận Q và K được chuẩn hóa bởi hệ số $\sqrt{d}$ (d là số chiều của *vector embedding*). Lý do cho việc chuẩn hóa này là do khi d có giá trị lớn, tích vô hướng của Q và K sẽ có giá trị lớn theo do đó, nếu không chuẩn hóa về giá trị nhỏ hơn thì hàm *softmax* sẽ có độ hội tụ khá chậm



Hình 2.5: Đồ thị hàm *softmax* và đạo hàm của một thành phần trong tập phân loại

Algorithm 1 Self_attention($context, w_K, w_Q, w_V$)

1: **Result:** Z
2: $K \leftarrow context \times w_K$
3: $Q \leftarrow context \times w_Q$
4: $V \leftarrow context \times w_V$
5: $Score \leftarrow QK^T$
6: $Z \leftarrow \frac{softmax(Score)}{\sqrt{d}}V$

Từ công thức tính của *self-attention*, ta có thể thấy rõ sự khác biệt giữa bộ mã hóa của *transformer* và bộ mã hóa của các mô hình *Seq2Seq* sử

dụng mạng hồi quy. Đối với mô hình hồi quy, dữ liệu đầu vào phải được mã hóa một cách tuần tự. Trong khi đối với *self-attention*, các từ trong văn bản có thể được mã hóa một cái song song và không bị phụ thuộc vào nhau. Nhờ đó mà có thể tăng được hiệu quả tính toán.

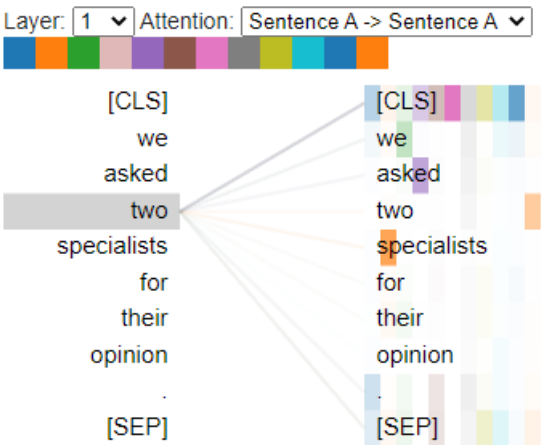
Cơ chế Multihead-attention

Đầu ra của *self-attention* cho ta biết được mối quan hệ của các từ trong câu dựa trên một góc nhìn nào đó. Bằng các xếp chồng nhiều lớp *self-attention* lại với nhau. Ta có thể biết được sự liên quan của các từ ngữ trong câu với nhiều gốc nhìn khác nhau. Từ đó mà có được thông tin đầy đủ hơn về câu cần dịch.

Cơ chế xếp chồng nhiều lớp *self-attention* lại với nhau được gọi là *Multihead attention*.

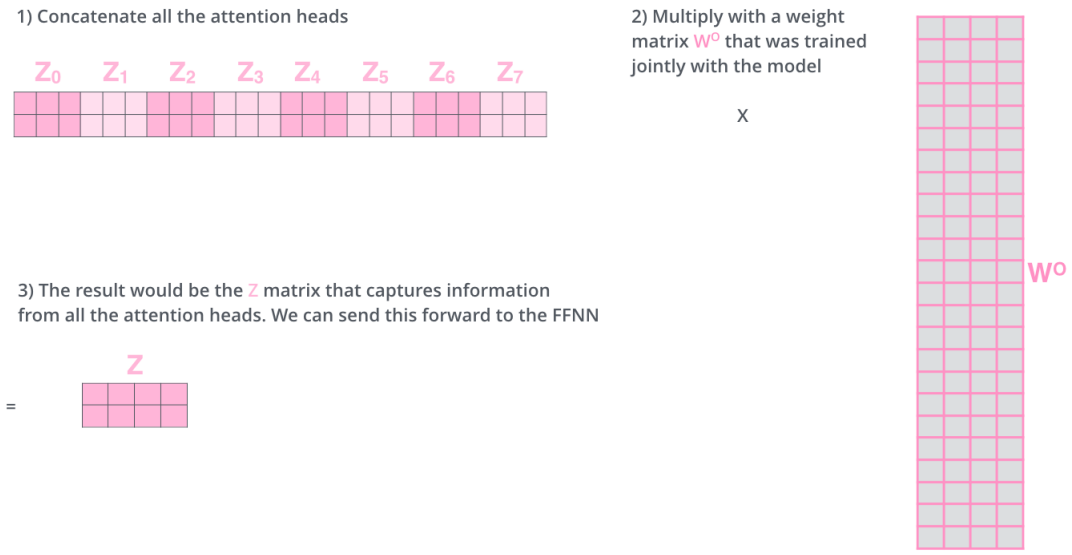
Ngoài ra, sử dụng cơ chế *self-attention* còn giúp ta tránh được trường hợp một từ phụ thuộc hoàn toàn vào chính nó. Ta mong muốn một phân bố xác suất quan hệ giữa một từ với các từ có ảnh hưởng đến nó.

Với việc sử dụng N lớp *self-attention* chạy song song với các bộ trọng số khác nhau, ta có được N ma trận *context* khác nhau. Lúc này, ta cần có một phương phát khác để tổng hợp thông tin từ các lớp *self-attention* này lại để đưa vào lớp *feed-forward*.



Hình 2.6: Minh họa cơ chế *Multihead attention*

Để làm được việc đó, *Transformer* ghép theo chiều ngang các ma trận *context* lại rồi nhân với một bộ trọng số để đưa ra một kết quả duy nhất là một ma trận với các dòng là các *context vector* cũng đồng thời là đầu vào cho bộ giải mã ở phía trên.

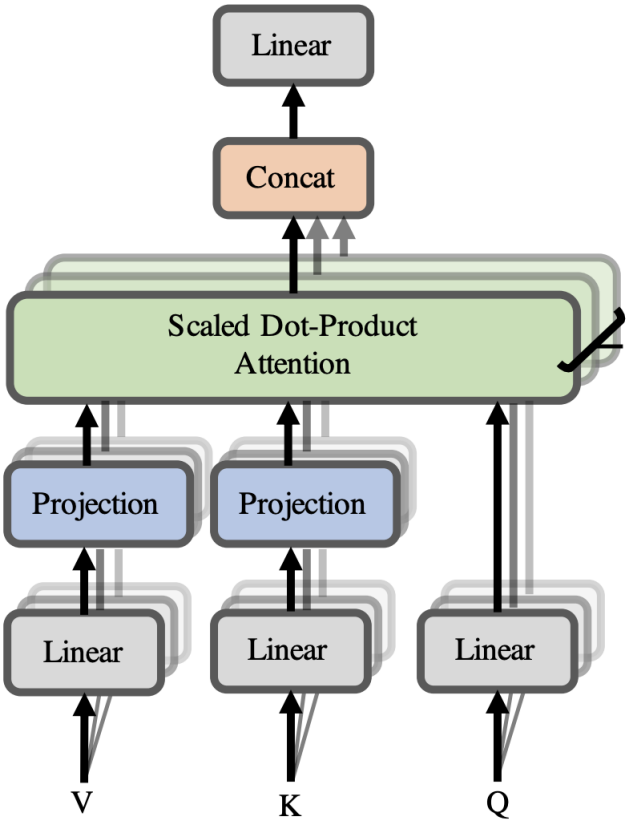


Hình 2.7: Cơ chế tổng hợp feed-forward trong Multihead attention[5]

Hình dưới minh họa cho việc sử dụng nhiều lớp *self-attention* trên cùng một câu với mỗi một màu tương ứng với kết quả của một lớp *self-attention* khác nhau.

Algorithm 2 Multihead attention(*context*)

1: **Result:** $Z \times w_O$
2: **for** $1...\#head$ **do**
3: $\quad Z_i \leftarrow self\_attention(context, wK_i, wQ_i, wV_i)$
4: $Z \leftarrow [Z_1, Z_2, ..., Z_{\#head}]$



Hình 2.8: Minh họa kiến trúc của *Multihead attention*[16]

Positional encoding

Cách tính toán song song của cơ chế *self-attention* dẫn đến một vấn đề đối với các từ trong đầu vào. *Embeedings* biểu diễn các từ này chưa biểu diễn được thông tin về thứ tự của các từ trong câu. Trong khi thông tin này là một thông tin quan trọng vì thay đổi vị trí của các từ có thể dẫn đến một câu hoàn toàn khác ý nghĩa.

Một ví dụ về việc đối chỗ một từ trong câu khiến cho ý nghĩa của câu trở nên trái ngược hoàn toàn:

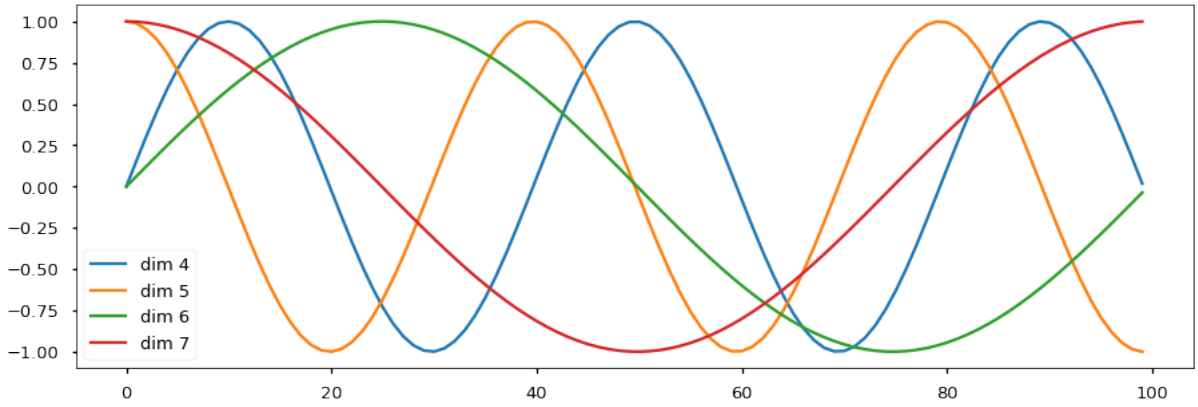
- "Please don't go! I love you!"
- "Please go! I don't love you!"

Transformer sử dụng cơ chế *positional encoding*. Cơ chế này giúp đưa thông tin về vị trí của các từ vào trong các *embeddings*. Cụ thể hơn, trước khi *embeddings* được đưa vào trong mô hình, nó được cộng với một vector tương ứng với vị trí trong câu. Những vector này tuân theo một một quy định nhất định. Chúng phải thể hiện được sự khác biệt về vị trí của các từ trong câu và cả khoảng cách của các từ trong câu.

Công thức tính toán các vector này như sau:

$$PE_{(pos,2i)} = \sin \frac{pos}{10000^{\frac{2i}{d_model}}}$$

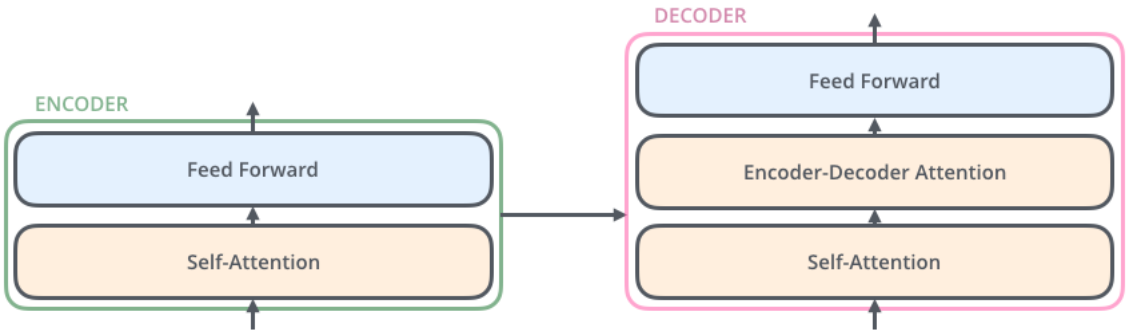
$$PE_{(pos,2i+1)} = \cos \frac{pos}{10000^{\frac{2i}{d_model}}}$$



Hình 2.9: Đồ thị positional embedding [13]

2.1.3 Lớp giải mã

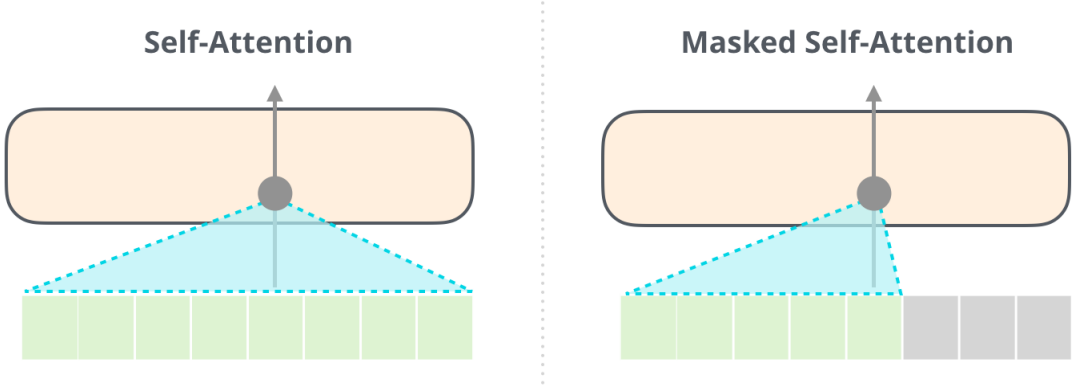
Các lớp của bộ giải mã cũng có 2 lớp con là *Self-attention* và *Feed-forwarding* giống với bộ mã hóa. Tuy nhiên giữa 2 lớp con này có một lớp trung gian là *Encoder-Decoder attention*. Lớp này có cơ chế giống với cơ chế *attention* trong mô hình *Seq2Seq* với thông tin của các *lớp ẩn (hidden layers)* chính là đầu ra của bộ mã hóa.



Hình 2.10: Minh họa kiến trúc bộ lớp mã hóa [5]

Masked self attention

Khác với lớp *self-attention* ở bộ mã hóa, lớp *masked self attention* ở bộ giải mã chỉ tìm kiếm các mối quan hệ của từ hiện tại với các từ đã được giải mã trước đó. Do đó, trước khi thực hiện tính toán trên hàm *softmax*, giá trị *score* của các từ chưa được giải mã sẽ được gắn nhãn là $-\infty$. Sau đó, các bước tính toán sẽ tương tự như trong bộ mã hóa.



Hình 2.11: So sánh *self attention* và *maked self attention*

Ta có thể điều chỉnh lại hàm *self attention* như sau:

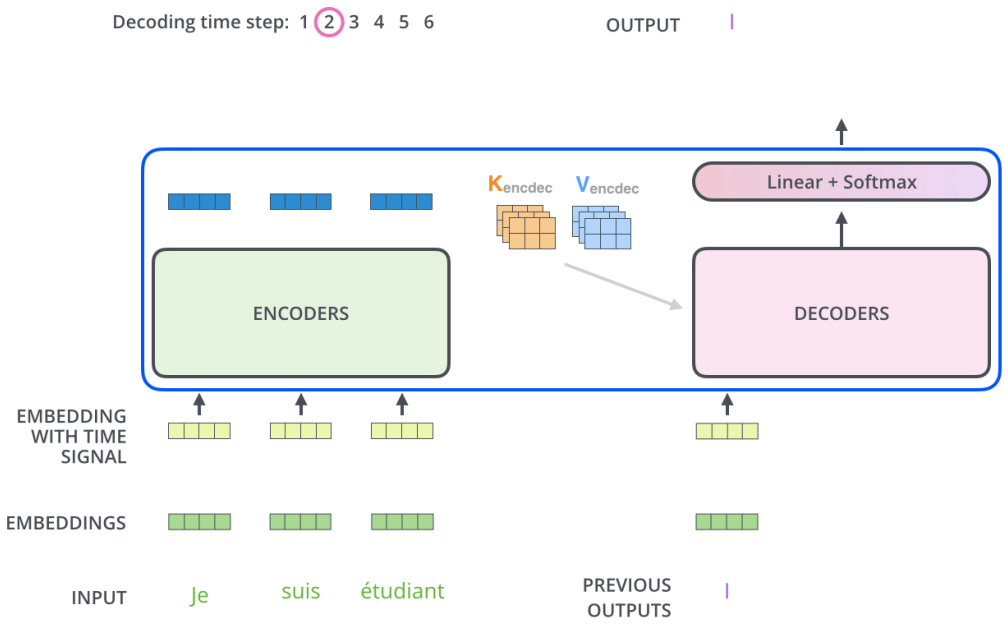
Algorithm 3 Self_attention($context, w_K, w_Q, w_V, masked$)

1: **Result:** Z
2: $K \leftarrow context \times w_K$
3: $Q \leftarrow context \times w_Q$
4: $V \leftarrow context \times w_V$
5: $Score \leftarrow QK^T$
6: **for** $i = masked + 1...len(context)$ **do**
7: Cột thứ i của $Score = -\infty$
8: $Z \leftarrow \frac{softmax(Score)}{\sqrt{d}}V$

Tham số *masked* thể hiện kể từ vị trí này, các phần tử phía sau của *score* sẽ được gán giá trị $-\infty$. Hàm *self_ attention* được gọi bởi hàm *multihead_ attention*. Do đó, ta cần truyền thêm tham số masked vào hàm này. Ta mặc định *masked = len(context)*, nhờ vậy mà mà cần truyền thêm tham số này khi gọi trong bộ mã hóa.

encoder-decoder attention

Một điểm khác biệt dễ thấy giữa bộ mã hóa và bộ giải mà là lớp *encoder-decoder attention* được đặt giữa 2 lớp *self attetion* và *feed-forward*. Công dụng của lớp này giúp cho lớp giải mã có thể giải mã ra được các từ của văn bản đích dựa trên độ liên quan của mỗi từ với các từ trong văn bản nguồn.



Hình 2.12: Minh họa cơ chế *encoder-decoder-atention* [5]

Đầu ra lớp trên cùng của mã hóa sẽ được biến đổi thành 2 vector K và V. Quá trình giải mã được thực hiện như sau:

Algorithm 4 decoder(context, pos)

1: **Result:** $feed\_forward(context)$
2: $context \leftarrow multihead\_attention(context, masked = pos)$
3: $context \leftarrow normalize(context)$
4: $context \leftarrow encoder\_decoder\_attention(K_{enc}, V_{env}, context)$

Để biểu thị cho mở đầu của đoạn văn bản và kết thúc của đoạn văn bản đến đánh dấu việc bắt đầu giải mã và kết thúc giải mã. *transformer* sử dụng 2 token đặc biệt nằm tách biệt với bộ từ vựng: *<bos> (begin of sentence)* và *<eos> (end of sentence)*.

Quá trình giải mã sẽ được bắt đầu với token *<bos>* và thực hiện một cách tuần từ cho đến khi giải mã ra token *<eos>*. Từ được giải mã ở bước trước sẽ được sử dụng làm đầu vào cho bước giải mã tiếp theo. Cụ thể thuật toán của toàn bộ quá trình giải mã như sau:

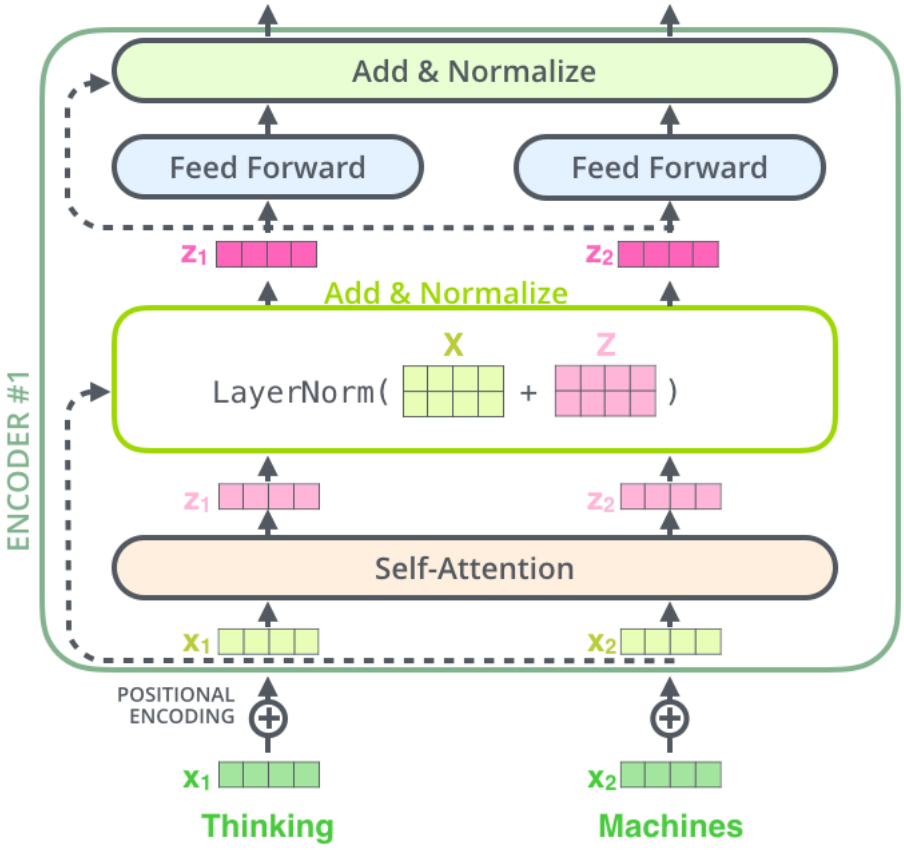
Algorithm 5 Quá trình giải mã

1: Khởi tạo $output_i \leftarrow < bos >$
2: Khởi tạo $i \leftarrow 1$
3: **do**
4: $\quad embed_i \leftarrow position\_encoding(output_{i-1})$
5: $\quad context \leftarrow embed_i$
6: $\quad$ **for do** $j = 1...\#layer$
7: $\quad\quad context \leftarrow decoder(context, masked = i)$
8: $\quad output_i \leftarrow softmax(context)$
9: **while** $output_i \neq < eos >$

2.1.4 residuals

Để tránh các trường hợp *Gradient biến mất*, *Transformer* kết hợp kiến trúc *residual* vào các lớp mã hóa. Từ đó mà thông tin từ các lớp trước có thể được sử dụng lại trong khi huấn luyện các lớp sau.

với mỗi lớp con trong cả bộ mã hóa lẫn bộ giải mã, ta đặt thêm một lớp *normalize* vào ngay trên lớp con đó. Lớp *normalize* này nhận đầu vào là kết quả của lớp con ngay dưới của nó và cả đầu vào trước khi đi vào lớp con đó. Thực hiện phép cộng hai ma trận này và chuẩn hóa lại trước khi đưa vào lớp con phía trên.



Hình 2.13: Minh họa kiến trúc *residual*[5]

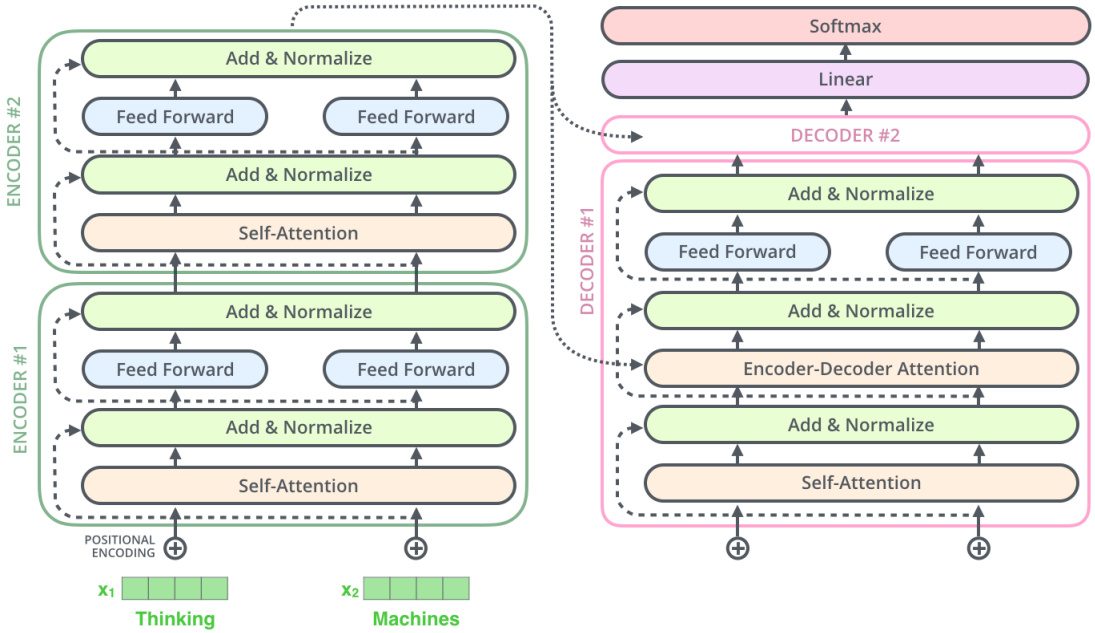
2.1.5 Tổng kết mô hình

Tổng kết lại, kiến trúc của *Transformer* bao gồm 2 thành phần chính là bộ mã hóa và bộ giải mã. Bộ mã hóa được tạo bởi nhiều lớp con gọi là các lớp mã hóa. Tương tự, bộ giải mã cũng được hình thành từ các lớp giải mã được xếp chồng lên nhau. Số lượng lớp con của bộ mã hóa bằng với số lượng lớp con của bộ giải mã.

Với mỗi lớp con của bộ mã hóa sẽ bao gồm một lớp multihead-attention và một lớp feed-forward. Ngoài ra còn áp dụng thêm kiến trúc residual block với các lớp normalize.

Với mỗi lớp giải mã, chúng được xếp chồng bởi 3 lớp con lần lượt là: *Masked Multihead Attention*, *Encoder Decoder Attention* và *Feed Forward-*

ing. Lớp giải mã cũng được áp dụng kiến trúc *Residual block* giúp mô hình tránh được hiện tượng *Gradient biến mất*



Hình 2.14: Tổng kết mô hình *Transformer* [5]

2.2 Mô hình WordGCN huấn luyện word embeddings cho kho ngữ liệu

2.2.1 Lý thuyết đồ thị cơ bản

Định nghĩa đồ thị

Đồ thị được định nghĩa là một cấu trúc rời rạc gồm các đỉnh và các cạnh nối các đỉnh đó. Ký hiệu biểu diễn một đồ thị như sau:

$$G = (V, E)$$

Trong đó, V là tập các *đỉnh (Vertices)* và E là tập các *cạnh (Edges)*. Các cạnh trong tập E được bởi diễn bởi một cặp (x,y) với $x, y \in V$. Số

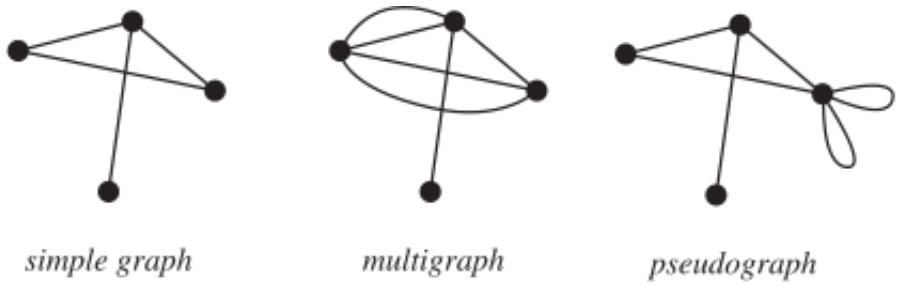
lượng đỉnh của đồ thị là $|V| = n$, còn số lượng cạnh của đồ thị là $|E| = m$

Các loại đồ thị

Đơn đồ thị (simple graph) là đồ thị thỏa $\forall x, y \in V|$ không tồn tại quá 1 cạnh nối giữa hai đỉnh này.

Đa đồ thị (multigraph) là đồ thị thỏa $\forall x, y \in V|$ có thể có nhiều hơn 1 cạnh nối giữa 2 đỉnh này

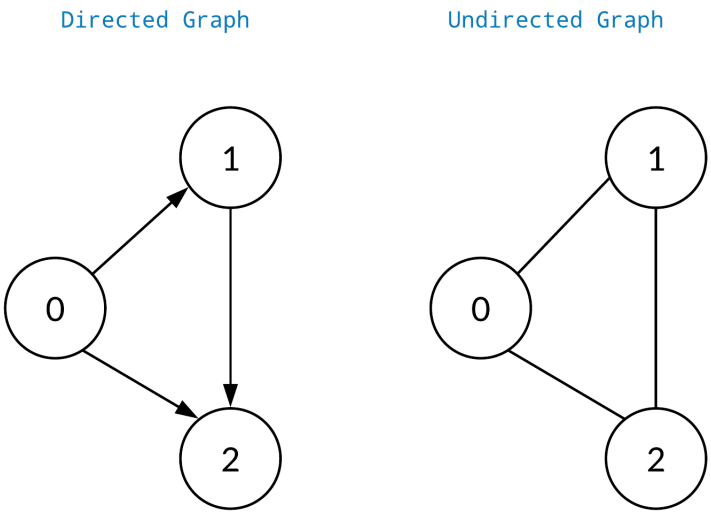
Giả đồ thị (speudograph) là đồ thị chứa các đỉnh có khả năng nối cạnh với chính nó.



Hình 2.15: Phân biệt đơn đồ thị, đa đồ thị và giả đồ thị

Đồ thị vô hướng (undirected graph) là đồ thị chứa các cạnh không định hướng. Nói cách khác, cạnh nối hai đỉnh x và y cũng sẽ được hiểu là cạnh đỉnh y với đỉnh x.

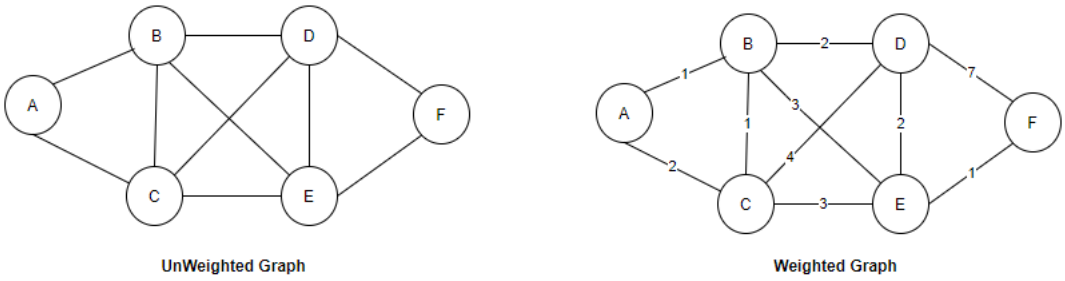
Đồ thị có hướng (directed graph) là đồ thị chứa các cạnh có hướng. nói cách khác, cạnh nói hai đỉnh x và y sẽ phân biệt với cạnh nối từ y đến x.



Hình 2.16: Phân biệt đồ thị vô hướng và đồ thị có hướng

Đồ thị có trọng số là đồ thị mà các cạnh của đồ thị được biểu diễn bằng một trọng số nào đó. Lúc này mỗi cạnh $e \in E$ sẽ được biểu diễn bởi (x, y, w). Với x và y là hai đỉnh của cạnh và w là trọng số của cạnh đó

Đồ thị không trọng số là đồ thị mà các cạnh của đồ thị không được biểu diễn bởi một trọng số.



Hình 2.17: Phân biệt đồ thị có trọng số và không trọng số

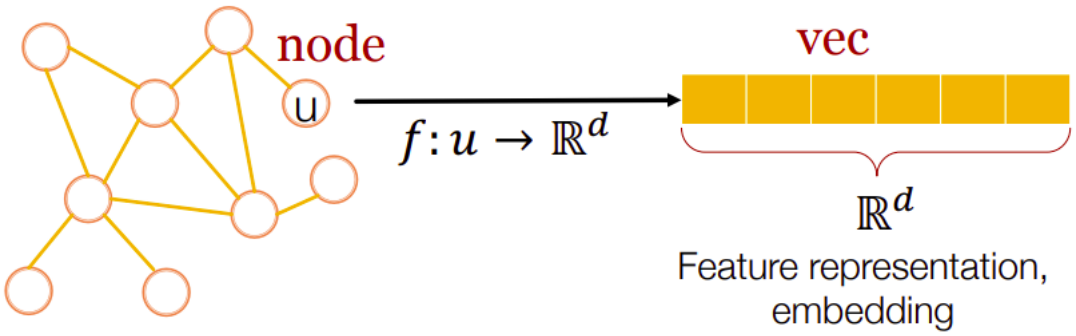
Ngoài ra, khi kết hợp cách tính chất của các loại đồ thị khác nhau, ta có thể có được các dạng đồ thị khác như: Đơn đồ thị vô hướng, Đa đồ thị có hướng, Giả Đa đồ thị có hướng có trọng số,..

Đồ thị còn có thể phân loại theo đồ thị vô hạn và độ thị hữu hạn, chỉ số lượng cạnh và đỉnh của đồ thị là vô hạn hay có thể đếm được.

Vector đặc trưng (feature vectors)

Lý thuyết đồ thị khi được áp dụng vào các bài toán học máy thường sẽ được bổ sung thêm khác khái niệm vector đặc trưng. Vector đặc trưng là những vector chỉ các đặc trưng của một đỉnh hoặc một cạnh của đồ thị. Các vector đặc trưng biểu diễn cho các đối tượng giống nhau phải có số lượng chiều như nhau.

Các vector này có thể có một cấu trúc đã được định nghĩa trước với tập các trường được khảo sát trên đối tượng tương ứng với đỉnh đang xét. Ngoài ra các vector của có thể được huấn luyện nhờ vào các mô hình học máy. Một bài toán được quan tâm đến khi nhắc đến các vector đặc trưng này là từ các liên kết của đồ thị và các *context vector* đã được định nghĩa trước, tìm ra các vector đặc trưng biểu diễn tốt cho các đối tượng đỉnh hoặc/và cạnh của đồ thị đó. Bài toán này được gọi là *representation learning*.



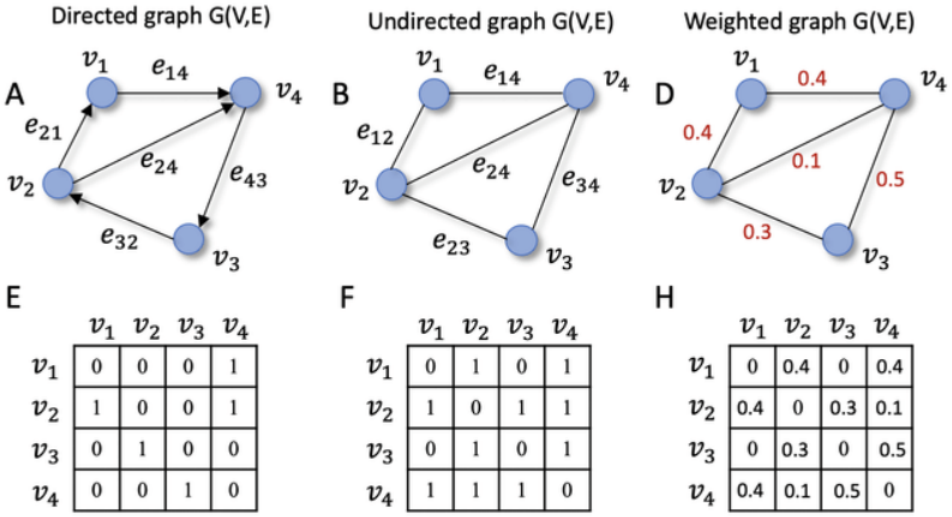
Hình 2.18: Minh họa về vector đặc trưng

Ma trận kề (adjacent matrix)

Ma trận kề là ma trận chứa các thông tin về các cạnh nối giữa các đỉnh trong đồ thị. Ma trận kề có kích thước $|V| \times |V|$. Phần tử ở hàng x cột y

của ma trận chứa thông tin về kết nối giữa 2 đỉnh u và v và có giá trị như sau:

- *Đồ thị không trọng số* : 0 hoặc x biểu thị hoặc không có cạnh nối giữa 2 điểm u và v hoặc có x cạnh đang nối giữa 2 đỉnh này.
- *Đồ thị có trọng số* : 0 hoặc w biểu thị hoặc không có cạnh nối giữa u và v hoặc có cạnh nối giữa u và v và trọng số của cạnh này là w.



Hình 2.19: Ma trận kề

Bậc của đồ thị

Bậc của một đỉnh được định nghĩa là tổng số cạnh nối với đỉnh đó. Đối với đơn đồ thị, số cạnh nối với đỉnh đang xét cũng bằng số đỉnh kề với đỉnh đang xét. Bậc của đỉnh u được ký hiệu là $deg(u)$.

Tính chất:

- Tổng số bậc của tất cả các đỉnh là số chẵn và có giá trị là $2m$ với m là số lượng cạnh của đồ thị:

$$\sum_{v \in V} deg(v) = 2m$$

- Tổng số đỉnh có bậc lẻ là số chẵn.
- Đối với đồ thị vô hướng, tổng giá trị của hàng x hoặc cột x của ma trận kề biểu diễn đồ thị đó chính bằng $deg(x)$

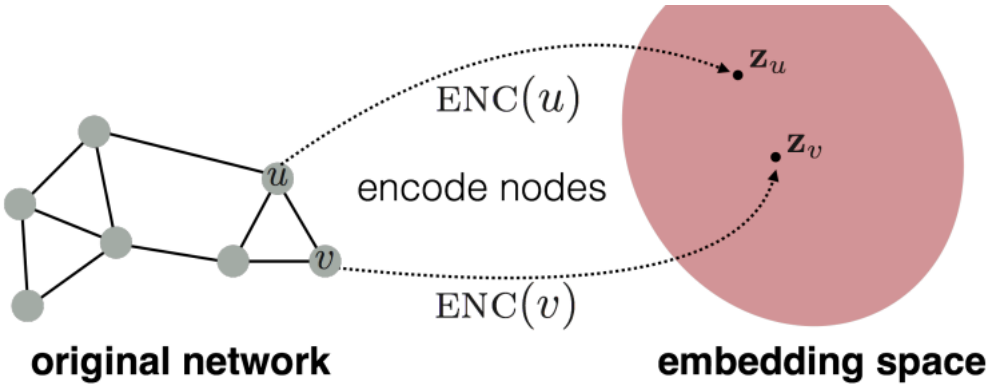
Đối với đồ thị có hướng, bậc của một đỉnh còn được phân ra làm 2 thành phần:

- Bán bậc vào là số lượng cạnh nối với đỉnh đang xét và có hường vào đỉnh đó. Ký hiệu bán bậc vào của đỉnh u là $deg^-(u)$
- Bán bậc ra là số lượng cạnh nối với đỉnh đang xét và có hường ra khỏi đỉnh đó. Ký hiệu bán bậc ra của đỉnh u là $deg^+(u)$

$$deg(u) = deg^-(u) + deg^+(u)$$

2.2.2 Mô hình Graph Convolution Network và bài toán representation learning

Để giải bài toán *representation learning* trên đồ thị, người ta đưa ra giả thuyết rằng, các nút trên đồ thị có khoảng cách càng gần nhau thì sẽ có các đặc tính giống nhau. Từ đó mà vector đặc trưng của những nút gần kế nhau cũng sẽ có khoảng cách gần nhau trong không gian latent.



Hình 2.20: Minh họa về bài toán *Representation learning*

Từ giả thuyết này, người ta đưa ra được ý tưởng chính của mạng *GCN* là sử dụng các vector đặc trưng của các nút kề với nút đang xét để học được đặc trưng của nút này.

Cho đồ thị $G = (V, E)$, có ma trận kề $A_{n\times n}$ và ma trận đặc trưng $X_{n\times d}$ với $n = |V|$ và d là chiều của ma trận đặc trưng. Mô hình *GCN* được kiến trúc bởi nhiều lớp ẩn chồng lên nhau. Gọi $H^{(l)}$ là đầu ra của lớp ẩn thứ l. $H^{(l)}$ là một ma trận có kích thước $n \times d_l$ với d_l là số lượng đặc trưng của từng nút tại lớp l. Công thức tính $H^{(l)}$ được biểu diễn một như sau:

$$H^{(l)} = \begin{cases} f(H^{(l-1)}, A), & l > 0 \\ X, & l = 0 \end{cases}$$

Trong đó, f là hàm tính đặc trưng của các nút dựa trên các lớp kề với nút đó. Do đó, f có biểu diễn như sau:

$$f(H^{(l)}, A) = \sigma(AH^{(l)}W^{(l)})$$

Với W là ma trận chứa các tham số học có kích thước $d_l \times d_{l+1}$. Và σ là hàm kích hoạt phi tuyến của mô hình.

Với công thức của hàm f như trên, ta thấy được 2 phần thiếu sót như sau:

- Vector đặc trưng của nút u chỉ được tính bởi đặc trưng của các nút kề nó mà chưa được xét bởi các đặc trưng nội tại của nút u. Ta có thể giải quyết vấn đề này bằng các thêm các *liên kết vòng (self loop)* từ một nút đến chính nó. Ta định nghĩa ma trận $\tilde{A}$ là ma trận kề của G đã bổ sung các liên kết vòng:

$$\tilde{A} = A + I$$

- Các đỉnh có bậc cao sẽ gây ảnh hưởng đến nhiều nút hơn các đỉnh khác. Và nó cũng sẽ được ảnh hưởng bởi nhiều đỉnh hơn các đỉnh

khác. Theo như công thức f ở trên, các đỉnh có bậc cao sẽ ma giá trị rất lớn hoặc rất nhỏ dẫn đến việc cập nhập gradient chậm lại trong quá trình lan truyền ngược. Để giải quyết vấn đề trên, *GCN* sử dụng phép chuẩn hóa *Symetric normalized Laplacian* để tránh sự thiên vị trong quá trình học. Từ đó mà ta có công thức đầy đủ của hàm f như sau:

$$H^{(l+1)} = \sigma(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{1/2}H^{(l)}W^{(l)})$$

Trong đó, $\tilde{D}$ là ma trận bậc của đồ thị G phép chuẩn hóa đối xứng laplacian có thể được hiểu là với mỗi $\tilde{A}_{u,v}$ sẽ được nhân với một lượng $\frac{1}{\sqrt{deg(u)}\sqrt{deg(v)}}$

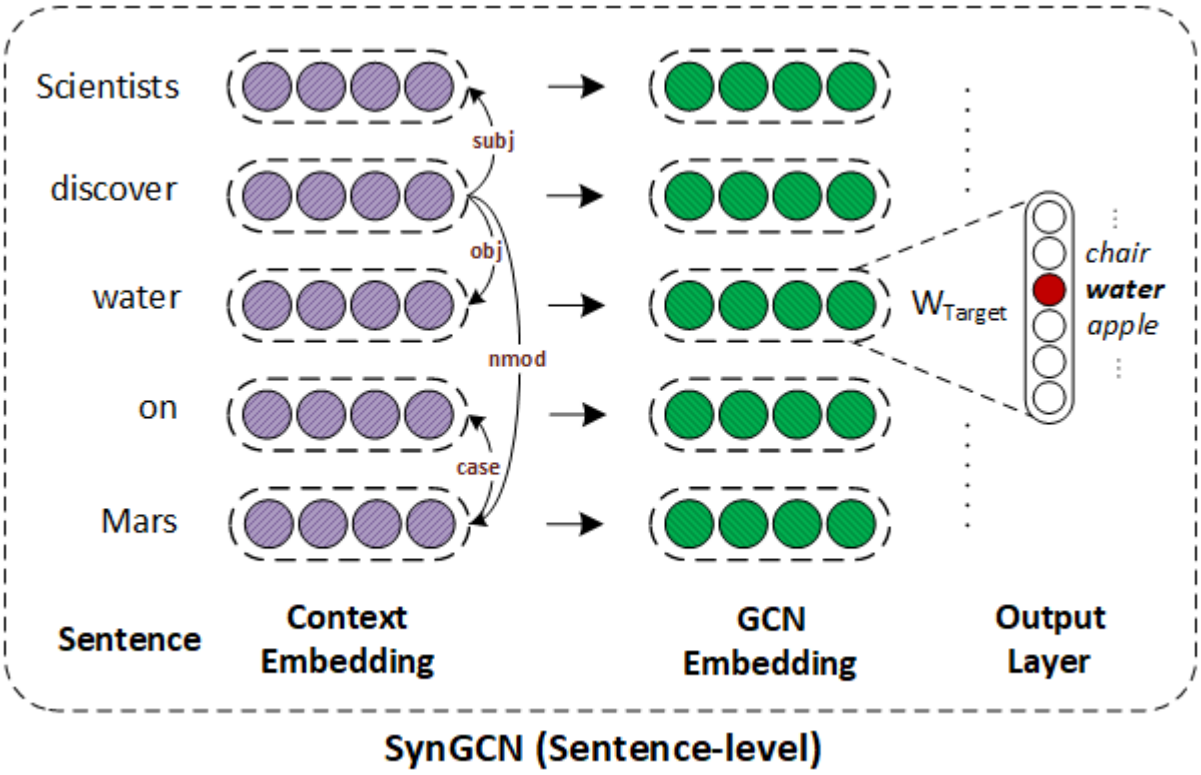
2.2.3 Mô hình WordGCN

WordGCN là mô hình được ghép bởi 2 mô hình con là *SynGCN* và *SemGCN*. Cả hai mô hình trên đều áp dụng kiến trúc *GCN* để trích xuất các *embeddings* của các từ trong kho ngữ liệu. Trong khi *SynGCN* trích xuất thông tin cú pháp (*syntactic context*) của các từ thông qua các liên kết của chúng trong các đoạn văn bản. Thì *SemGCN* rút trích các đặc trưng về ngữ nghĩa (*semantic context*) của các từ thông qua các mối quan hệ như: đồng nghĩa, trái nghĩa, từ viết tắt,...

Mô hình *SynGCN*

SynGCN quan tâm đến đặc trưng của một từ thông qua ví trị và vai trò của từ đó trong câu. Theo *Universal Dependency* [12] Mỗi từ trong câu sẽ có một,mmối quan hệ với một từ khác trong câu hoặc là chủ thể của cả câu (không có liên kết đến từ khác). Các mối quan hệ đó được gọi là các *Universal Dependencies*. Từ đây, ta có thể xem mỗi câu $s = \{w_1, w_2, ..., w_n\}$ trong kho ngữ liệu là một đồ thị con dạng cây có hướng $G_s = (V_s, E_s)$ với các nút là các từ $V_s = \{w_1, w_2, w_3..., w_n\}$ và các *dependency* là các cạnh của đồ thị có dạng (w_i, w_j, d_{ij}) với d_{ij} là mối quan

hệ giữa w_i và w_j. Lúc này, ta có thể áp dụng Thuật toán *GCN* lên đồ thị này.



Hình 2.21: Mô hình cơ bản của *SynGCN* [14]

Nhận xét thấy *GCN* áp dụng vào trường hợp này khá tương tự với mô hình *Continues Bag of Word(CBOW)*. Mô hình này sử dụng tập các từ xung quanh w_i để học các đặc trưng của w_i. Tập các từ như vậy được gọi là tập *Context* của w_i.

$$C_{w_i} = \{w_{i+j} : -c \leq j \leq c, j \neq 0\}$$

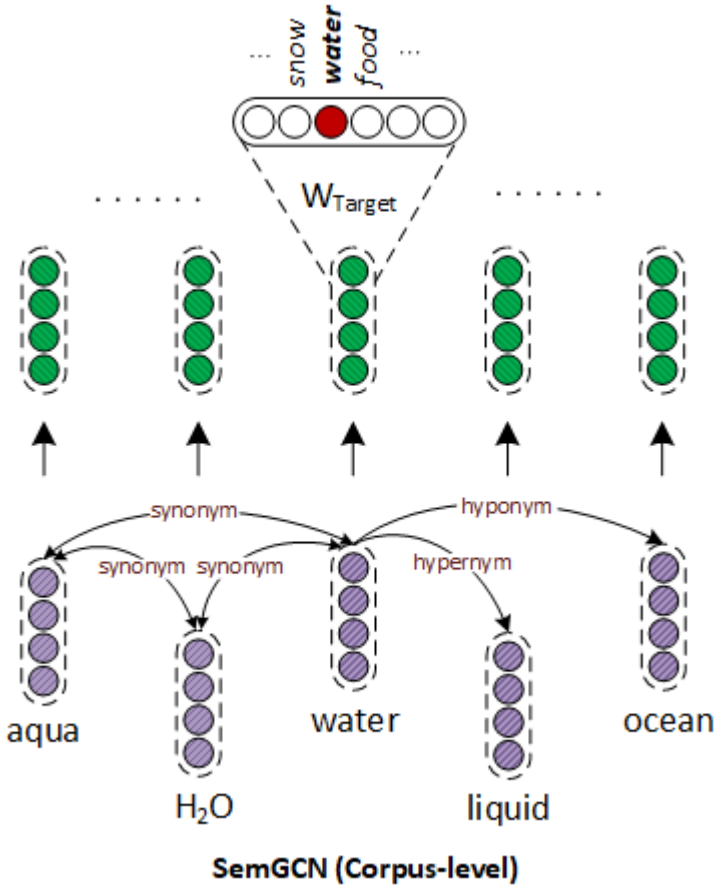
Trong đó, c là kích thước của cửa sổ trượt.

Đối với *SynGCN*, tập *Context* của w_i được định nghĩa là tập các các từ tương ứng với nút kề với w_i:

$$C_{w_i} = N(w_i) = \{w_j : (w_i, w_j, l_{i,j}) \in E_s\}$$

Mô hình SemGCN

Các *embedding* sau khi được huấn luyện qua mạng SynGCN sẽ được đưa vào mạng *SemGCN* để học các đặc trưng về ngữ nghĩa (*semantic context*). Mô hình cho phép học các mối quan hệ có tính đối xứng lẫn các mối quan hệ không đối xứng. Đối với các mô hình trước đây, hoặc là chúng chỉ có thể giải quyết các mối quan hệ đối xứng hoặc chưa xử lý tốt với các môi không đối xứng. Gọi $G = (V, E)$ là đồ thị được tạo thành từ các ngữ nghĩa trong bộ dữ liệu. Ta sẽ có các đỉnh là các từ trong bộ nghữ liệu còn các cạnh là các mối quan hệ giữa chúng. Các cạnh trong G có định hướng. Đối với 2 từ x và y có liên kết đối xứng thì giữa 2 đỉnh tương ứng với x và y sẽ có 2 cạnh ngược hướng.



Hình 2.22: Ý tưởng cơ bản của *SemGCN* [14]

Sau khi có được đồ thì biểu diễn, ta có thể sử dụng thuật toán *GCN* để tối ưu các *embeddings* đã được huấn luyện từ mô hình *SynGCN*.

Chương 3

Phương pháp đề xuất

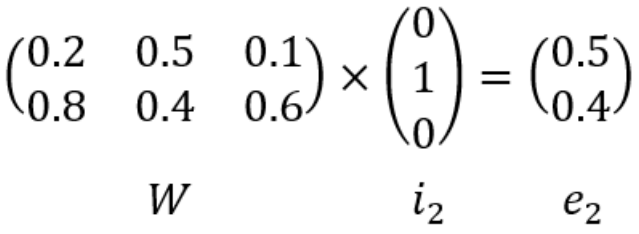
3.1 Áp dụng mô hình WordGCN vào mô hình Transformer

Ở mô hình đề xuất, sinh viên muốn thay thể module mã hóa *embedding* đầu vào và cả mã hóa *embedding* đầu ra bằng các *embedding* được huấn luyện thuông qua mô hình *WordGCN*.

Ở mô hình đề xuất *Transformer* [15], lớp mã hóa embedding đầu vào và đầu ra được cài đặt bằng một lớp feed forward. Ban đầu, các token được mã hóa bằng phương pháp *onehot encoding*. Sau đó, được mã hóa thành các embedding của kích thước d = 512. Gọi W có kích thước $n_{voc} \times d$ là ma trận trọng số của lớp mã hóa embedding đầu vào và đầu ra. Với n_{voc} là số lượng từ vựng trong kho ngữ liệu và d là số chiều của embedding. Embedding tương ứng với từ thứ i trong kho ngữ liệu được tính bằng công thức:

$$e_x = i_x \times W$$

Trong đó, e_x là *embedding* của từ thứ x còn i_x là *vector one-hot encoding* của từ thứ x.



Hình 3.1: Minh họa lớp mã hóa embedding với số lượng từ vựng trong kho ngữ liệu $n_{voc} = 3$ và số chiều của embedding $d = 2$

Các tiếp cận này chưa khai thác được các đặc trưng về cú pháp và các đặc trưng về ngữ nghĩa của các từ vựng trong một ngôn ngữ. Bằng cách sử dụng WordGCN, các embeddings sẽ được huấn luyện trước qua 2 mô hình *SynGCN* và *SemGCN*. Kết quả sẽ được lưu vào trong một tập tin. Sau đó được truyền truyền thẳng vào mô hình *transformer* .

(Hình minh họa quá trình)

3.2 Chuẩn bị dữ liệu

3.2.1 Tập dữ liệu WMT14

WMT14 (Workshop on Statistical Machine Translation 2014) [7] là một workshop chia sẻ về các bài toán được quan tâm trong lĩnh vự dịch máy. Trong đó, có 4 bài toán:

- Translation task
- Quality estimation task
- metric task
- Medical translation task

Trong phạm vi của khóa luận, sinh viên quan tâm đến bài toán dịch máy (translation task) với 2 ngôn ngữ là tiếng Anh và tiếng Đức.

Với tập dữ liệu huấn luyện, WMT14 đã cung cấp khoảng 4.5 triệu câu từ 3 kho ngữ liệu song ngữ Anh-Đức:

- Kho ngữ liệu song ngữ *Europarl*: Được trích xuất từ các thủ tục hành chính của nghị viện Châu Âu
- Kho ngữ liệu song ngữ *News Commentary*: bao gồm bình luận chính trị và kinh tế được thu thập từ trang web Project Syndicate
- Kho ngữ liệu song ngữ *Common Crawl*: bao gồm các thông tin được thu thập từ tổng hợp các nguồn trên internet.

Với tập dữ liệu kiểm thử, dữ liệu được tổng hợp từ các mẫu truyện từ internet. Theo [7], tập kiểm thử bao gồm 1500 câu tiếng Anh được dịch sang tiếng Đức và 1500 câu tiếng Đức được dịch về tiếng anh. Các mẫu dịch được cung cấp bởi các chuyên gia từ *Capita*.

Bảng 3.1: Bảng thống kê số liệu của kho ngữ liệu song ngữ *Europarl*

	German ↔ English	
Sentences	1,920,209	
Words	50,486,398	53,008,851
Distinct words	381,583	115,966

Bảng 3.2: Bảng thống kê số liệu của kho ngữ liệu song ngữ *New Commentary*

Bảng 3.3:

	German ↔ English	
Sentences	201,288	
Words	5,105,101	5,046,157
Distinct words	150,760	65,520

Bảng 3.4: Bảng thống kê số liệu của kho ngữ liệu song ngữ *Common Crawl*

	German ↔ English	
Sentences	2,399,123	
Words	54,575,405	58,870,638
Distinct words	1,640,835	823,480

3.2.2 Tiền xử lý dữ liệu cho mô hình SynGCN

Định dạng của dữ liệu

Theo [10], dữ liệu đầu vào của *SynGCN* phải được tiền xử lý thành các tập tin có cấu trúc như sau:

- *voc2id.txt*: Gán id cho mỗi từ trong kho từ vựng

$$voc2id : W \rightarrow \mathbb{R}$$

- *id2freq.txt*: Thống kê tần số xuất hiện của các từ vựng không kho ngữ liệu
- *de2id.txt*: Gán id cho các mối quan hệ trong universal dependencies
- *data.txt*: Biểu diễn lại kho ngữ liệu theo định dạng:

$$N\ M\ tok_1\ tok_2\ tok_3...tok_N\ dep_1\ dep_2....dep_M$$

  - N là số lượng từ trong câu và M là số lượng mối quan hệ giữa các từ trong câu.
  - tok_i là id của từ thứ i trong câu.
  - dep_i là mối quan hệ thứ i trong câu. Mỗi mối quan hệ được biểu diễn dưới dạng:

$$source\_token|destination\_token|dep\_rel\_label$$

Trong đó, *source_token* và *destination_token* là hai từ phát sinh mối quan hệ với nhau. *dep_rel_label* là mối quan hệ giữa hai từ này.

Voc2id

WMT14 đã cung cấp sẵn cho ta tập V là tập hợp các từ vựng trong kho ngữ liệu. Do đó, ta chỉ cần đánh số thứ tự cho các từ trong V.

Algorithm 6 Tiền xử lý dữ liệu voc2id

Khởi tạo $tok_{<unk>} = 0$
Khởi tạo $count \leftarrow 1$
for $w \in V$ **do**
    $tok_w = count$
    $count \leftarrow count + 1$

id2freq

Algorithm 7 Tiền xử lý dữ liệu id2freq

Khởi tạo $\forall w \in V : freq_w = 0$
Khởi tạo $freq_{<unk>} = 0$
for $w \in W$ **do**
    if $w \in V$ **then**
        $freq_w \leftarrow freq_w + 1$
    else
        $freq_{<unk>} \leftarrow freq_{<unk>} + 1$

Trong đó, $<unk>$ là các từ lạ không nhận diện được từ bộ từ vựng.

de2id

Tập các mối quan hệ *Dep* được tổng hợp từ *Universal dependencies*.

Algorithm 8 Tiền xử lý dữ liệu de2id

1: **for** $relation \in Dep$ **do**
2: $dep_{relation} = count$
3: $count \leftarrow count + 1$

data

Theo *WordGCN*, để có thể rút trích các mối quan hệ của các từ trong câu, ta sử dụng công cụ *CoreNLP Parser*. Ở bước này, do các câu có thể xử lý một các độc lập nhau. Ta có thể chia tập dữ liệu ra thành các tập nhỏ hơn để xử lý giúp tăng hiệu quả tính toán và giảm gánh nặng tài nguyên. Các phần cứng chạy song song sẽ chia sẻ với nhau chung các tập tin *voc2id*, và *de2id*. Mỗi máy thành phần thứ i sẽ được xử lý tập $S_i \subset S$ với S là tập tất cả các câu trong kho ngữ liệu.

Algorithm 9 Tiền xử lý dữ liệu data

1: $document = CoreNLP\_Parser(S_i)$
2: **for** $sentence \in document$ **do**
3: Ghi số lượng từ trong câu vào data.txt
4: Ghi số lượng quan hệ trong câu vào data.tx
5: **for** $word \in sentence$ **do**
6: **if** $word \in V$ **then**
7: Ghi tok_{word} vào data.txt
8: **else**
9: Ghi $tok_{<unk>}$ vào data.txt
10: **for** $dep \in dependencies(sentence)$ **do**
11: Ghi $dep_{src}|dep_{dest}|dep_{relation}$ vào data.txt
12: Writeln

3.2.3 Tiền xử lý dữ liệu cho mô hình SemGCN

WordNet

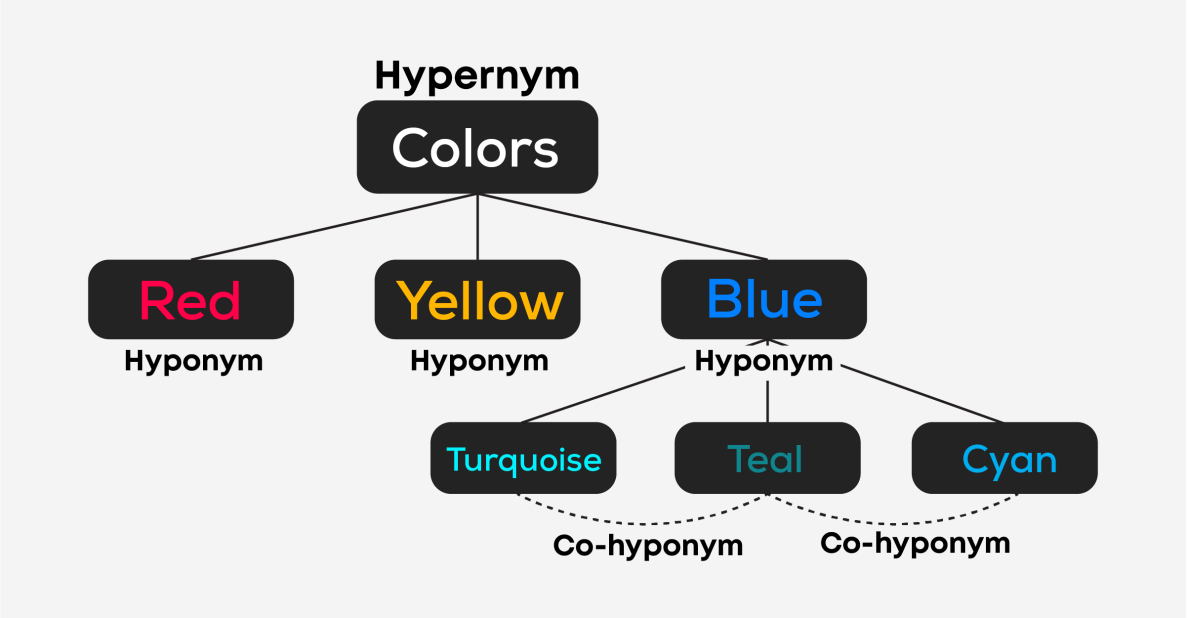
WordNet là kho ngữ liệu từ vựng gom nhóm các từ tương đồng thành các tập gọi là các synsets. Các từ trong một synset có thể được thay đổi

cho nhau mà không làm thay đổi ý nghĩa của câu ở một số trường hợp nhất định.

Các mối quan hệ được đề cập trong WordNet chủ yếu là thượng-hạ vị. Ngoài ra còn cung cấp thêm các cặp từ trái nghĩa.

Khái niệm hạ vị chỉ một từ hoặc một cụm từ có ngữ nghĩa được kế thừa từ một từ khác. Từ đó, ta có khái niệm thượng vị là một từ có ngữ nghĩa chứa đựng ngữ nghĩa của một từ khác. Có thể thấy, thượng hạ vị là các mối quan hệ không đối xứng, và không thể có một từ vừa là thượng vị cũng vừa là hạ vị của một từ. Từ đó, ta có thể tạo ra được một rừng (tập hợp của các đồ thị con dạng cây) có hướng. Với các đỉnh cha là thượng vị của các đỉnh con.

Ngoài ra, ta còn có khái niệm đồng hạ vị (co-hyponym) chỉ các cặp từ cùng là hạ vị của một từ.



Hình 3.2: Ảnh minh họa thượng vị và hạ vị

Theo như ví dụ trên, các màu sắc "red" (đỏ), "yellow" (vàng) và "blue" (xanh dương) là các hạ vị của "colors" (màu sắc) do chúng đều có ý nghĩa là một màu sắc. "Colors" được gọi là thượng vị của "red", "yellow" và

"blue". Bên cạnh đó, "yellow" và "blue" do có chung một thượng vị nên chúng được gọi là đồng thượng vị.

Định dạng dữ liệu

Theo [10], *SemGCN* quan tâm đến 4 mối quan hệ ngữ nghĩa và được xác định bởi 4 tập tin sau:

- *Antonyms*: bao gồm các cặp từ có ý nghĩa trái ngược nhau trong kho từ vựng. Mối quan hệ trái nghĩa là mối quan hệ 2 chiều và sẽ biểu diễn bằng 2 cạnh đối nhau trong đồ thị ngữ nghĩa.
- *Hypernyms*: chứa các cặp bao gồm một từ và thượng vị của nó. Mối quan hệ này là mối quan hệ một chiều.
- *Hyponyms*: chứa các cặp từ bao gồm một từ và một hạ vị của nó. Mối quan hệ này cũng là moios quan hệ một chiều.
- *Synonyms*: bao gồm nhiều dòng, với mỗi dòng biểu diễn một từ và tập các từ đồng nghĩa với từ đó.

Dựa trên *WordGCN*, các mối quan hệ *antonyms, hypernym* và *hyponym* được trích xuất nhờ vào WordNet, còn mối quan hệ *synonym* thì được trích xuất dựa vào PPDB (The paraphrase database).

3.2.4 Tiền xử lý dữ liệu cho mô hình transformer

Dữ liệu đầu vào của *transformer* bao gồm 2 phần:

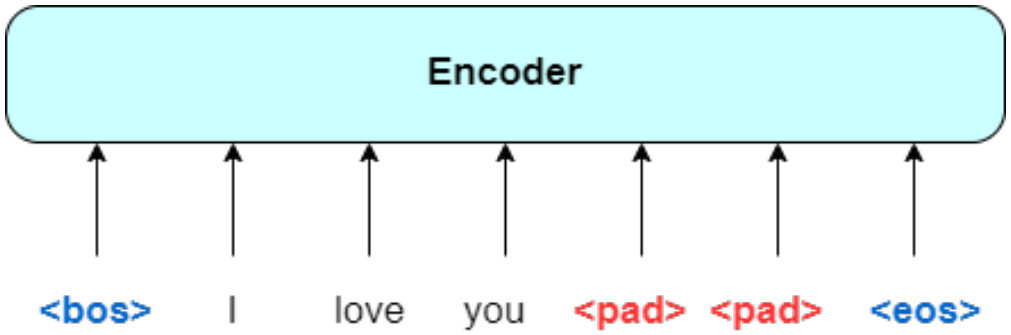
- kho ngữ liệu WMT14 bao gồm các cặp câu song ngữ Anh-Đức.
- Các embedding được huấn luyện bởi mô hình WordGCN

Với dữ liệu WMT14, ta cần thực hiện tokenization các từ trong kho ngữ liệu với id của các từ phải tương ứng với id trong tập tin voc2id.txt. Ngoài ra, ta phải bổ sung thêm ba token đặc biệt vào vào bộ từ vựng như sau:

- $<pad>$: Ma trận chứa dữ liệu đầu vào của *transformer* có kích thước $n \times len\_max$. Trong đó n là số lượng câu còn len_max là kích thước của câu có chiều dài dài nhất trong kho ngữ liệu. Do đó, dẫn đến một số hàng của ma trận sẽ không có giá trị. Ta bổ sung thêm một token đặc biệt này vào để lắp đầy các vị trí đó. Embedding tương ứng với $<pad>$ sẽ được gán giá trị bằng với vector 0.

- $<bos>$: Token này dùng để đánh dấu bắt đầu của một câu. Giá trị của Embedding tương ứng sẽ được gán là vector với toàn giá trị 1 ở mọi chiều.

- $<eos>$: token này dùng để đánh dấu kết thúc của một câu. Giá trị của Embedding tương ứng sẽ được gán là vector với toàn giá trị -1 ở mọi chiều.



Hình 3.3: Minh họa về các token đặc biệt trong mô hình *transformer*

Dữ liệu các embedding đã được huấn luyện bởi mô hình WordGCN sẽ được lưu trữ trong một tập tin có dạng như sau:

```
good 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
great 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
awesome 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
```

Hình 3.4: Định dạng của tập tin chứa các *embedding* được huấn luyện bởi mô hình *WordGCN*. Ở ví dụ này mỗi từ được biểu diễn bởi một embedding của số chiều là 20.

Mỗi dòng sẽ chứa thông tin về *embedding* của một từ trong kho ngữ liệu. Trong đó:

- Một từ đầu dòng thuộc kho ngữ liệu.
- d số tiếp theo biểu thị giá trị của *vector embedding* của từ đang xét.

Chương 4

Huấn luyện và Kết quả

4.1 Huấn luyện

Như đã trên bày ở trên, cả 2 mô hình *WordGCN* và *Transformer* đều sẽ được huấn luyện bởi kho ngữ liệu *WMT14*. Với khoảng 4.5 triệu câu song ngữ Anh Đức. Sử dụng bộ tự vựng bao gồm 50000 từ cho cả 2 ngôn ngữ Anh và Đức.

4.1.1 Kích thước batch của dữ liệu huấn luyện

Kích thước batch của dữ liệu được huấn luyện đối với mô hình *SynGCN* là 128 và mô hình *SemGCN* là 64.

Đối với *Transformer*, mỗi batch dữ liệu sẽ có kích thước 25000.

Thuật toán tối ưu

SynGCN và *SemGCN* sử dụng thuật toán *Adam* với giá trị của *learning rate*:

$$l_{rate} = 10^{-3}$$

Đối với *Transformer*, thuật toán tối ưu được sử dụng vẫn là *Adam* với

các siêu tham số như sau:

$$\begin{cases} l_{rate} = 10^{-3} \\ \beta_1 = 0.9 \\ \beta_2 = 0.98 \\ \epsilon = 10^{-9} \end{cases}$$

4.1.2 Kết quả

Mô hình	BLEU
transformer (baseline)	27.3
WordGCN + *transformer*	26.5

Bảng 4.1: Bảng kết quả huấn luyện của mô hình cơ sở và mô hình đề xuất

Kết quả huấn luyện của mô hình đề xuất tuy chưa thể vượt qua được kết quả của mô hình cơ sở tuy nhiên vẫn có hiệu quả khá tốt so với các mô hình trước đây. Nguyên nhân của kết quả này có thể là do các nguyên nhân sau:

- Các bộ siêu tham số chưa được tối ưu giá trị.
- Kho từ vựng còn thiếu sót nhiều từ.
- Kích thước embedding được huấn luyện ở mô hình *WordGCN* chưa phù hợp khi đưa vào mô hình *transformer* .

Chương 5

Kết luận

Qua khóa luận này, sinh viên đã nghiên cứu về các thuật toán học sâu trên các bộ dữ liệu non-Euclidean (Graph Neural Network). Cụ thể hơn là mô hình WordGCN, được cấu thành bởi 2 mô hình con là *SynGCN* và *SemGCN*. Cả 2 mô hình con đều được lấy ý tưởng từ một biến thể thường gặp của Graph Neural Network đó là Graph Convolutional Network.

Ngoài ra, sinh viên còn nghiên cứu về bài toán học máy. Hướng nghiên cứu là về mô hình *transformer* . Một mô hình dịch máy dựa hoàn toàn vào cơ chế attention, bỏ qua các kiến trúc kinh điển như mạng tích chập hay mạng hồi quy.

Cuối cùng, sinh viên đề xuất sử dụng các word embedding được huấn luyện từ mô hình WordGCN để áp dụng vào mô hình *transformer* với mục đích sử dụng các thông tin về cú pháp và ngữ nghĩa từ các embedding này làm tăng hiệu năng của mô hình *transformer* .

Định hướng phát triển. Kết quả tuy chưa đạt được như kì vọng, song đã đống góp được ý tưởng để phát triển các thuật toán xử lý ngôn ngữ tự nhiên trong tương lai. Có thể thấy tiềm năng của việc sử dụng các word embedding được huấn luyện bởi các mô hình non-Euclidean có thể áp dụng cho không chỉ các thuật toán dịch máy mà còn cho các thuật toán khác. Một số bài toán đang được quan tâm như:

- Phân tích các bình luận

- Các bài toán về gợi ý phim ảnh, bài hát, sản phẩm,...
- Bài toán tự động điền của các công cụ tìm kiếm như Google, Bing,...
- Bài toán phát hiện email rác, gợi ý trả lời email.

Tài liệu tham khảo

Tiếng Anh

[4] ai, d2l. *Sequence to Sequence Learning*. URL: https://colab.research.google.com/github/d2l-ai/d2l-en-colab/blob/master/chapter_recurrent-modern/seq2seq.ipynb#scrollTo=l8L11XZknvp8 (visited on 07/24/2022).

[5] Alammar, Jay. *The Illustrated Transformer*. URL: https://jalammar.github.io/illustrated-transformer/ (visited on 07/24/2022).

[6] Bahdanau, Dzmitry, Cho, Kyunghyun, and Bengio, Yoshua. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *arXiv e-prints*, arXiv:1409.0473 (Sept. 2014), arXiv:1409.0473. arXiv: 1409.0473 [cs.CL].

[7] Bojar, Ondrej et al. "Findings of the 2014 Workshop on Statistical Machine Translation". English. In: *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, June 2014, pp. 12–58.

[8] datadriveninvestor. *How do lstm networks solve the problem of vanishing gradients*. URL: https://medium.com/datadriveninvestor/how-do-lstm-networks-solve-the-problem-of-vanishing-gradients-a6784971a577 (visited on 07/24/2022).

[9] Ethnologue.com. *How many languages are there in the world?* URL: https://www.ethnologue.com/guides/how-many-languages (visited on 07/24/2022).

[10] (IISc), MALL Lab. *WordGCN*. URL: https://github.com/malllabiisc/WordGCN (visited on 07/24/2022).

[11] Lin, Tianyang et al. *A Survey of Transformers*. 2021. DOI: 10.48550/ARXIV.2106.04554. URL: https://arxiv.org/abs/2106.04554.

[12] Nivre, Joakim et al. "Universal Dependencies v2: An Evergrowing Multilingual Treebank Collection". In: *arXiv e-prints*, arXiv:2004.10643 (Apr. 2020), arXiv:2004.10643. arXiv: 2004.10643 [cs.CL].

[13] Rush, Alexander. *The Annotated Transformer*. URL: http://nlp.seas.harvard.edu/2018/04/03/attention.html (visited on 07/24/2022).

[14] Vashishth, Shikhar et al. "Incorporating Syntactic and Semantic Information in Word Embeddings using Graph Convolutional Networks". In: *Proceedings of the 57th Conference of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 3308–3318. URL: https://www.aclweb.org/anthology/P19-1320.

[15] Vaswani, Ashish et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by Guyon, I. et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[16] Weng, Lilian. *Attention? Attention!* URL: https://lilianweng.github.io/posts/2018-06-24-attention/ (visited on 07/24/2022).