

## MỤC LỤC

<b>MỞ ĐẦU.....</b>	<b>1</b>
<b>CHƯƠNG 1. CƠ SỞ LÍ THUYẾT.....</b>	<b>4</b>
1.1 Đồ thị .....	4
1.1.1 Định nghĩa .....	4
1.1.2 Các dữ liệu dạng đồ thị.....	4
1.1.3 Các loại tác vụ với đồ thị .....	6
1.2 Mạng nơ-ron đồ thị .....	6
1.2.1 Mạng nơ-ron đồ thị cơ bản .....	6
1.2.2 Bộ tự mã hóa đồ thị .....	7
1.2.3 Mạng nơ-ron đồ thị không-thời gian.....	8
1.2.4 Các kiến trúc mạng nơ-ron đồ thị khác .....	8
1.3 Cơ chế chú ý .....	9
1.3.1 Giới thiệu .....	9
1.3.2 Cơ chế chú ý trong học sâu.....	9
1.3.3 Các loại cơ chế chú ý.....	11
1.4 Mô hình Transformer .....	12
1.4.1 Giới thiệu .....	12
1.4.2 Bộ mã hóa .....	12
1.4.3 Bộ giải mã.....	13
1.4.4 Lớp chú ý .....	14
1.4.5 Mạng nơ-ron truyền thẳng theo vị trí.....	15
1.4.6 Mã hóa theo vị trí .....	16

1.5 Biểu diễn mã hóa hai chiều từ Transformers.....	16
1.5.1 Giới thiệu .....	16
1.5.2 Kiến trúc BERT .....	17
1.5.3 Tiền huấn luyện BERT .....	17
1.5.4 Tinh chỉnh BERT .....	18
<b>CHƯƠNG 2. CÁC MÔ HÌNH MẠNG NƠ-RON ĐỒ THỊ TRONG BÀI TOÁN XÁC THỰC THÔNG TIN .....</b>	<b>20</b>
2.1 Mạng tích chập đồ thị.....	20
2.2 Mạng đồ thị cơ chế chú ý .....	21
2.3 Mô hình GraphSAGE .....	23
2.3.1 Thuật toán sinh embedding .....	23
2.3.2 Học các tham số của GraphSAGE .....	24
2.3.3 Các kiến trúc hàm tổng hợp.....	24
2.4 Mạng tích chập đồ thị hai chiều.....	25
2.4.1 Xây dựng đồ thị lan truyền và phân tán .....	26
2.4.2 Tính toán các biểu diễn nút bậc cao.....	27
2.4.3 Tăng cường đặc trưng nút gốc.....	27
2.4.4 Các biểu diễn sự lan truyền và sự phát tán để phân loại tin tức .....	28
2.5 Mạng nơ-ron đồ thị với học liên tục .....	28
<b>CHƯƠNG 3. THÍ NGHIỆM HỆ THỐNG PHÁT HIỆN TIN GIẢ UPFD VÀ KẾT QUẢ .....</b>	<b>30</b>
3.1 Hệ thống phát hiện tin giả dựa trên sở thích người dùng UPFD .....	30
3.1.1 Tổng quan.....	30
3.1.2 Mã hóa sở thích nội sinh.....	30

3.1.3 Trích xuất ngữ cảnh ngoại sinh.....	31
3.1.4 Tổng hợp thông tin .....	32
3.2 Hàm mätt mát trong bài toán phát hiện tin giả.....	32
3.3 Thiết lập thí nghiệm .....	32
3.3.1 Tập dữ liệu .....	32
3.3.2 Chỉ số đánh giá.....	33
3.3.3 Các mô hình được sử dụng .....	34
3.3.4 Chi tiết triển khai thí nghiệm.....	34
3.4 Kết quả thí nghiệm .....	35
3.4.1 So sánh hiệu năng .....	35
3.4.2 Nghiên cứu sự cắt bỏ .....	35
<b>KẾT LUẬN.....</b>	<b>39</b>
<b>CHỈ MỤC.....</b>	<b>40</b>
<b>TÀI LIỆU THAM KHẢO.....</b>	<b>44</b>



## DANH MỤC HÌNH VẼ

Hình 1.1	Ví dụ biểu diễn phân tử bằng đồ thị [18]. . . . .	5
Hình 1.2	Ví dụ tác vụ cấp cạnh với đồ thị [18]. . . . .	7
Hình 1.3	Cơ chế chú ý [25]. . . . .	10
Hình 1.4	Kiến trúc Transformer [26]. . . . .	13
Hình 1.5	Cơ chế chú ý trong Transformer [26]. . . . .	14
Hình 1.6	Quá trình tiền huấn luyện và tinh chỉnh BERT [27]. . .	17
Hình 1.7	Biểu diễn đầu vào BERT [27]. . . . .	18
Hình 2.1	Hai cơ chế chú ý trong GAT [24]. . . . .	22
Hình 2.2	Các mô hình lan truyền trong GCN [32]. . . . .	26
Hình 2.3	Mô hình BiGCN. GCL là lớp GCN (GCN layer) [32]. .	27
Hình 3.1	Hệ thống phát hiện tin giả UPFD [37]. . . . .	31
Hình 3.2	Hiệu năng phát hiện tin giả trên các biến thể UPFD khác nhau. . . . .	38

## **DANH MỤC BẢNG BIỂU**

Bảng 3.1	Thống kê về tập dữ liệu. . . . .	33
Bảng 3.2	Ma trận nhầm lẫn. . . . .	34
Bảng 3.3	So sánh hiệu năng phát hiện tin giả giữa các mô hình.	36
Bảng 3.4	Hiệu năng phát hiện tin giả trên các bộ mã hóa văn bản khác nhau. . . . .	37

## DANH MỤC TỪ VIẾT TẮT

- BERT** Bidirectional Encoder Representations from Transformers (Biểu diễn mã hóa hai chiều từ Transformers)
- BiGCN** Bi-Directional Graph Convolutional Network (Mạng tích chập đồ thị hai chiều)
- EWC** Elastic Weight Consolidation (Hợp nhất trọng số đòn hồi)
- GAE** Graph Autoencoder (Bộ tự mã hóa đồ thị)
- GAT** Graph Attention Network (Mạng đồ thị cơ chế attention)
- GCN** Graph Convolutional Network (Mạng tích chập đồ thị)
- GEM** Gradient Epsiodic Memory (Bộ nhớ phân đoạn gradient)
- GNN** Graph Neural Network (Mạng neuron đồ thị)
- GNNCL** Graph Neural Network with Continual Learning (Mạng neural đồ thị với học liên tục)
- GraphSAGE** Graph SAmple and aggreGatE (Mô hình lấy mẫu và tổng hợp đồ thị)
- LSMT** Long Short-Term Memory (Bộ nhớ ngắn hạn dài)
- NLP** Natural Language Processing (Xử lí ngôn ngữ tự nhiên)
- RNN** Recurrent Neural Network (Mạng neural hồi quy)
- STGNN** Spatial-Temporal Graph Neural Network (Mạng neural đồ thị không-thời gian)
- UPFD** User Preference-aware Fake News Detection (Hệ thống phát hiện tin giả dựa trên sở thích người dùng)

# MỞ ĐẦU

## Bài toán xác thực thông tin

Sự phát triển của các mạng xã hội trong thời đại công nghiệp 4.0 đã góp phần vào sự bùng nổ thông tin trên Internet, và chúng cũng trở thành nguồn tin tức chính cho chúng ta cập nhật hàng ngày. Tuy nhiên, sử dụng mạng xã hội là một con dao hai lưỡi khi mà thông tin trên mạng rất khó kiểm chứng về độ xác thực, trong đó việc lan truyền tin giả (fake news) có thể gây ra sự nhận thức sai của người dân về tình hình trong nước và thế giới, đe dọa trật tự, an toàn xã hội và an ninh quốc gia.

Tin giả có thể được hiểu là tin tức sai sự thật được tạo ra và lan truyền nhằm mục đích lừa đảo hoặc xuyên tạc tình hình thực tế. Ta có thể thấy rất nhiều trường hợp tin giả trong cuộc sống như tin tức về dịch bệnh Covid-19, đời sống thường nhật, chính trị trong và ngoài nước, v.v. từ các nguồn thông tin không chính thống. Tuy nhiên, các nền tảng mạng xã hội hiện nay (facebook, twitters, youtube, v.v.) chưa tích hợp công nghệ phát hiện và chặn tin giả sớm, dẫn tới việc những quảng cáo sản phẩm không đúng công dụng thực tế (đặc biệt là thuốc và thực phẩm chức năng) hay tin tức bịa đặt về cuộc sống, xuyên tạc lịch sử được phát tán tràn lan trên các nền tảng này. Do đó, việc phát hiện và ngăn chặn kịp thời tin giả trước khi chúng lan truyền rộng là một nhiệm vụ hết sức cấp thiết.

## Các phương pháp hiện có để giải quyết bài toán xác thực thông tin

Các phương pháp xác thực thông tin hay nói cách khác là phương pháp phát hiện tin giả có thể được phân thành bốn nhóm [1]: phương pháp dựa trên nội dung (content-based), phương pháp dựa trên ngữ cảnh (context-based), phương pháp dựa trên lan truyền (propagation-based) và phương pháp dựa trên học đa nhãn (multilabel learning-based).

Phương pháp dựa trên nội dung bao gồm phương pháp hai phương pháp con: dựa trên tri thức (knowledge-based) và dựa trên văn phong (style-based). Phương pháp dựa trên tri thức sẽ sử dụng từ một tập tri thức thật (true knowledge) để xác định một tin tức là thật hay giả [2]. Trong khi đó, phương pháp dựa trên văn phong xác định tin giả từ văn phong khác biệt của chúng nhằm thu hút sự quan tâm của độc giả và trở nên nổi bật hơn tin thông thường [3].

Phương pháp dựa trên ngữ cảnh xác định độ uy tín của nguồn phát tán thông tin, trong đó độ uy tín được xác định từ sự phản hồi của người dùng và độ tin cậy của bản thân tin tức [4].

Phương pháp dựa trên lan truyền phát hiện tin giả bằng cách trích xuất thông tin liên quan đến sự phát tán tin tức và người chia sẻ chúng [5].

Phương pháp dựa trên học đa nhãn cũng tương tự phương pháp dựa trên nội dung, nhưng một tin tức sẽ có nhiều nhãn *{thật, giả}* được gán cho các đặc trưng của tin tức thay vì chỉ một nhãn *thật* hoặc *giả* [6]. Điều này phù hợp với trên thực tế do tin giả có thể không hoàn toàn sai sự thật mà trong đó bao gồm cả thông tin thật và thông tin giả.

Ngoài ra, khi ta kết hợp hai phương pháp đã nêu, ví dụ như nội dung- ngữ cảnh hoặc lan truyền-nội dung, ta sẽ có phương pháp kết hợp (hybrid). Phương pháp này hiện nay được sử dụng nhiều do chúng có thể bắt được nhiều thông tin có ý nghĩa hơn liên quan tới tin giả, từ đó nâng cao hiệu quả của mô hình phát hiện tin giả.

Trong các phương pháp trên, mô hình lõi được sử dụng phổ biến nhất là Mạng nơ-ron đồ thị (Graph Neural Network - GNN) . GNN là một kĩ thuật áp dụng các thuật toán học sâu trên kiến trúc đồ thị [7] và là mô hình được sử dụng phổ biến nhất hiện nay trong phát hiện tin giả nhờ một số ưu điểm. Thứ nhất, GNN có thể kết hợp các phương pháp dựa trên thông tin nội dung, lan truyền và ngữ cảnh tin tức [8]. Thứ hai, GNN có thể đạt được hiệu quả tương đương hoặc tốt hơn các mô hình truyền thống mà không cần đến thông tin về văn bản [9]. Thứ ba, GNN có tính linh hoạt trong việc định nghĩa propagation pattern nhờ các bước nhảy ngẫu nhiên (random walks) và bộ tổng hợp lặp (iterative aggregators) [10].

Bên cạnh phát hiện tin giả, GNN còn được áp dụng thành công trong nhiều bài toán khác nhau như phát hiện vật thể (object detection) [11], [12], phân tích cảm xúc (sentiment analysis) [13], [14] và dịch máy (machine translation) [15], [16]. Sự phát triển nhanh chóng của các mô hình GNN cho phát hiện tin giả nhờ sự phổ biến rộng rãi của các mạng xã hội, cụ thể là sự tăng lên về số lượng người dùng, tin tức được đăng tải và tương tác giữa các người dùng. Kết quả là, các mạng xã hội đã trở thành các kiến trúc đồ thị phức tạp và gây khó khăn cho các thuật toán phát hiện tin giả. Phát hiện và ngăn chặn

tin giả trên mạng xã hội vẫn là một thách thức lớn cần được giải quyết. Các nguyên nhân trên đã thúc đẩy em nghiên cứu Một số mô hình ngôn ngữ và bài toán xác thực thông tin.

# CHƯƠNG 1. CƠ SỞ LÍ THUYẾT

Phần này trình bày cơ sở lí thuyết liên quan tới đồ thị, GNN, các mô hình GNN cơ bản, các kĩ thuật và mô hình liên quan tới xử lý ngôn ngữ tự nhiên trong bài toán xác thực thông tin. Các kiến thức này là nền tảng để xây dựng các mô hình GNN sẽ được trình bày trong Chương 2.

## 1.1 Đồ thị

Trong bài toán xác thực thông tin hay phát hiện tin giả trên mạng xã hội, ta cần một công cụ để biểu diễn sự tương tác giữa các người dùng với nhau cũng như với các bài đăng. Những sự tương tác lẫn nhau này có thể được biểu diễn hiệu quả bằng kiến trúc đồ thị, trong đó các người dùng / bài đăng được biểu diễn bằng *nút* đồ thị và các tương tác qua lại được biểu diễn bằng *cạnh* đồ thị. Mục 1.1 sẽ trình bày các khái niệm liên quan tới đồ thị và giới thiệu một số dữ liệu dạng đồ thị cùng với các tác vụ liên quan tới đồ thị.

### 1.1.1 Định nghĩa

Một đồ thị (graph)  $G = \{V, E\}$  bao gồm tập các nút (node/vertex)  $V = \{v_1, \dots, v_n\}$  và tập các cạnh (edge)  $E = \{e_{11}, \dots, e_{nm}\}$ , trong đó  $e_{ij} = (v_i, v_j)$  là cạnh nối hai nút liền kề  $v_i$  và  $v_j$ . Đồ thị được biểu diễn bằng một ma trận kề  $\mathbf{A} \in \{0, 1\}^{n \times n}$  với:

$$\mathbf{A}_{ij} = \begin{cases} 1, & e_{ij} \in E, \\ 0, & e_{ij} \notin E. \end{cases} \quad (1.1)$$

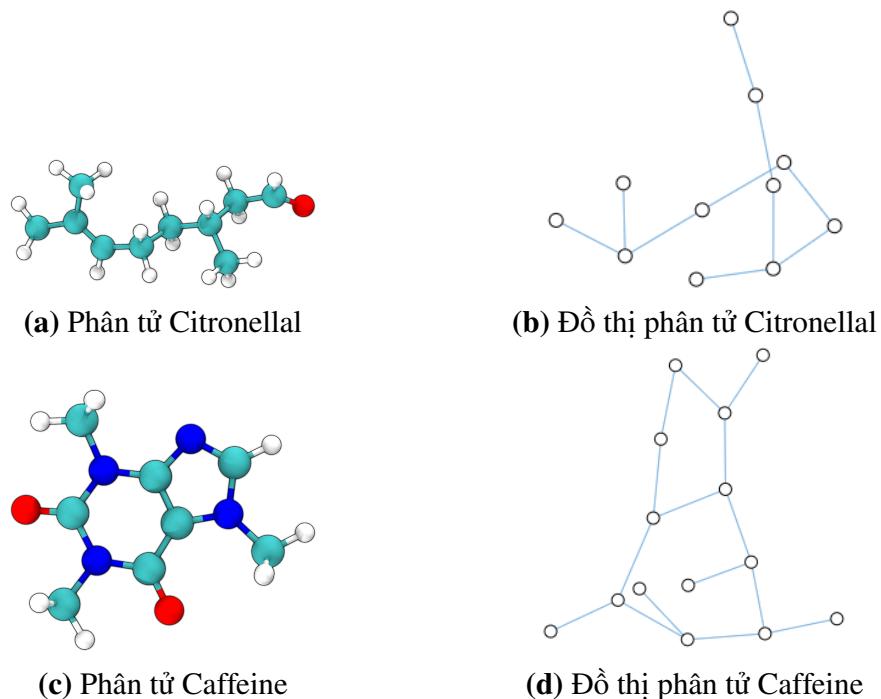
Đồ thị có thể được phân loại dựa theo các cạnh, bao gồm đồ thị vô hướng và đồ thị có hướng. Đồ thị vô hướng không phân biệt giữa nút nguồn và nút đích của một cạnh, do đó  $\mathbf{A}$  là ma trận đối xứng. Trong khi đó, các cạnh trong đồ thị có hướng xác định một nút nguồn và một nút đích, và ma trận  $\mathbf{A}$  là không đối xứng.

### 1.1.2 Các dữ liệu dạng đồ thị

#### a, Phân tử

Các phân tử là thứ tạo nên vật chất, và chúng được tạo nên từ các nguyên tử và electrons. Đồ thị là một công cụ hiệu quả để biểu diễn các phân tử, trong đó các nút là các nguyên tử và các cạnh là các liên kết giữa chúng [17]. Hình

1.1 thể hiện hai ví dụ phân tử Citronellal và Caffeine được biểu diễn bằng đồ thị.



**Hình 1.1:** Ví dụ biểu diễn phân tử bằng đồ thị [18].

### b, Mạng xã hội

Các mạng xã hội chứa rất nhiều thông tin về hành vi, thói quen, sở thích của cá nhân và tổ chức. Ta có thể xây dựng một đồ thị biểu diễn các nhóm người bằng cách mô hình hóa các cá nhân là các nút và quan hệ giữa họ là các cạnh của đồ thị.

### c, Trích dẫn

Các bài báo khoa học luôn có trích dẫn đến các bài báo khác. Các trích dẫn này có thể được biểu diễn bằng một đồ thị, trong đó mỗi bài báo là một nút và mỗi trích dẫn từ bài báo này đến bài báo khác là một cạnh *có hướng*.

### d, Các ví dụ khác

Đồ thị còn được dùng trong bài toán gán nhãn vật thể, trong đó các vật thể là các nút và quan hệ giữa chúng là các cạnh. Các mô hình học máy, mã nguồn lập trình [19] và phương trình toán học [20] cũng có thể được biểu diễn bằng đồ thị, trong đó các biến là các nút và các toán tử là các cạnh có nút nguồn và nút đích.

Kiến trúc của các đồ thị thực tế có thể rất khác nhau tùy vào loại dữ liệu, có đồ thị có rất nhiều nút nhưng rất ít cạnh và ngược lại. Các tập dữ liệu đồ

thị có thể khác nhau về số lượng nút, cạnh và tính kết nối (connectivity).

### 1.1.3 Các loại tác vụ với đồ thị

Có ba loại tác vụ mà ta có thể thực hiện với đồ thị: cấp đồ thị, cấp nút, và cấp cạnh [18].

#### a, Tác vụ cấp đồ thị

Trong tác vụ cấp đồ thị (graph-level), ta cần đưa ra dự đoán cho toàn bộ đồ thị. Ví dụ, với một phân tử, ta muốn dự đoán mùi của phân tử đó, hoặc liệu nó có liên kết với một thụ thể gây bệnh hay không. Bài toán này tương tự với bài toán phân loại ảnh, trong đó ta muốn gán một nhãn cho toàn bộ bức ảnh, hay bài toán phân tích cảm xúc văn bản, trong đó ta muốn xác định cảm xúc của toàn bộ câu văn.

#### b, Tác vụ cấp nút

Trong tác vụ cấp nút (node-level), ta cần đưa ra dự đoán cho mỗi nút trong đồ thị. Ví dụ, ta muốn dự đoán mỗi người trong một mạng xã hội ủng hộ ai trong cuộc bầu cử tổng thống. Bài toán này tương tự với bài toán phân vùng ảnh, trong đó ta muốn gán nhãn mỗi pixel trong một bức ảnh. Còn với văn bản, một bài toán tương tự là dự đoán từ loại trong câu (ví dụ danh từ, động từ, trạng từ, ...).

#### c, Tác vụ cấp cạnh

Trong tác vụ cấp cạnh (edge-level), ta cần dự đoán quan hệ giữa các nút trong đồ thị. Hình 1.2 là một ví dụ phân tích ngữ cảnh ảnh, trong đó bên cạnh xác định các vật thể trong ảnh, ta còn muốn dự đoán quan hệ giữa chúng.

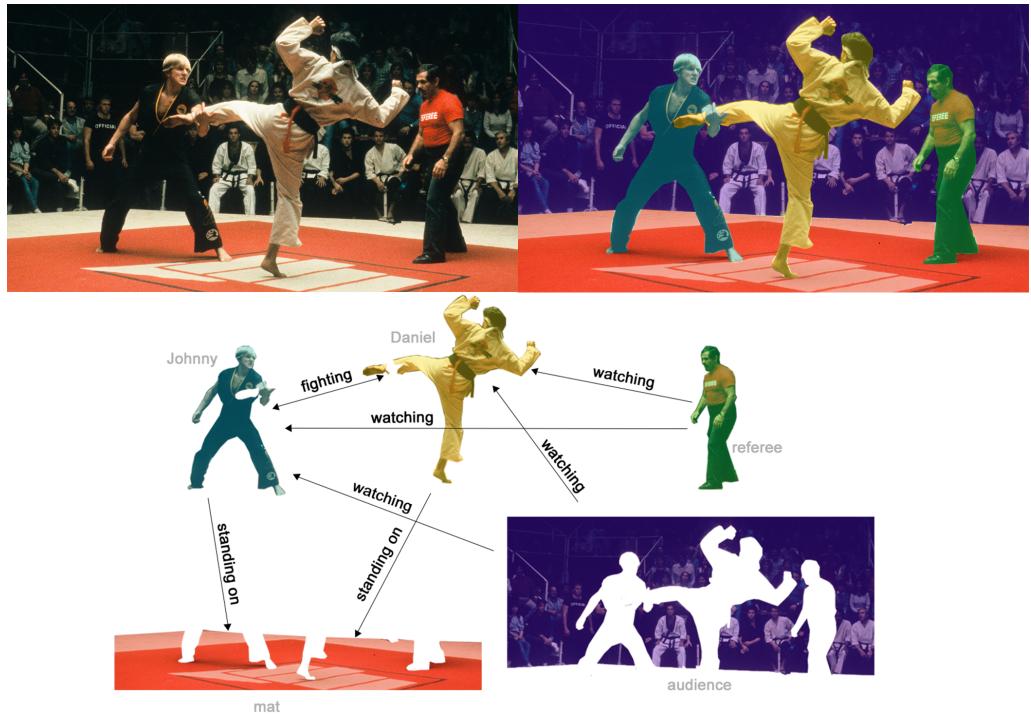
## 1.2 Mạng nơ-ron đồ thị

Mô hình mạng nơ-ron chuyên biệt để học từ dữ liệu dạng đồ thị được gọi là mạng nơ-ron đồ thị. Mục 1.2 sẽ trình bày một số kiến trúc mạng nơ-ron đồ thị cơ bản được sử dụng phổ biến.

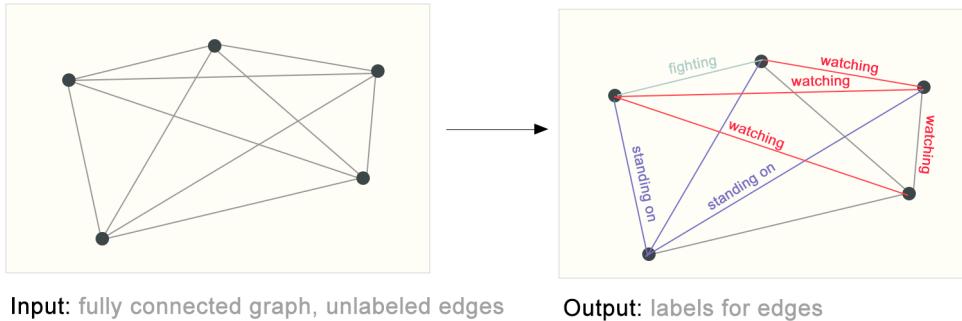
### 1.2.1 Mạng nơ-ron đồ thị cơ bản

Giả sử ta có một đồ thị  $G = \{V, E\}$  với  $N$  nút và  $M$  cạnh;  $\mathbf{A} \in \mathbb{R}^{N \times N}$  là ma trận kề và ma trận đặc trưng  $\mathbf{X} \in \mathbb{R}^{N \times D}$  với  $D$  là số chiều đặc trưng của nút. GNN sẽ dùng  $\mathbf{A}$  và  $\mathbf{X}$  làm đầu vào và có đầu ra tại lớp mạng  $(l + 1)$  là:

$$\mathbf{H}^{(l+1)} = f(\mathbf{H}^{(l)}, \mathbf{A}), \quad (1.2)$$



(a) Quan hệ giữa các vật thể trong ảnh



(b) Đồ thị quan hệ giữa các vật thể trong ảnh

**Hình 1.2:** Ví dụ tác vụ cấp cạnh với đồ thị [18].

trong đó  $f$  là một hàm lan truyền và  $\mathbf{H}^{(0)} = \mathbf{X}$ . Một dạng đơn giản của hàm lan truyền là  $f(\mathbf{H}^{(l)}, \mathbf{A}) = \sigma(\mathbf{A}\mathbf{H}^{(l)}\mathbf{W}^{(l)})$  với  $\sigma$  là một hàm kích hoạt phi tuyến (ví dụ hàm ReLU) và  $\mathbf{W}^{(l)}$  là ma trận trọng số lớp  $l$ .

Giả sử GNN có  $L$  lớp mạng, thì đầu ra tại lớp  $L$  là  $\mathbf{H}^{(L)} = \mathbf{Z} \in \mathbb{R}^{N \times F}$  với  $\mathbf{Z}$  là đầu ra cấp nút của GNN và  $F$  là số chiều đặc trưng đầu ra của nút.

### 1.2.2 Bộ tự mã hóa đồ thị

Bộ tự mã hóa đồ thị (Graph Autoencoder - GAE) [21] là một kiến trúc mạng nơ-ron với hai thành phần: 1) bộ mã hóa (encoder) chuyển các nút trên đồ thị thành một không gian véc-tơ các đặc trưng ẩn; và 2) bộ giải mã (decoder) giải mã thông tin trên đồ thị từ các véc-tơ đặc trưng ẩn. Trong GAE,

hàm  $f$  được định nghĩa là:

$$f(\mathbf{H}^{(l)}) = \sigma(\tilde{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}), \quad (1.3)$$

trong đó  $\mathbf{A} = \varphi(\mathbf{Z}\mathbf{Z}^T)$  là ma trận kè khôi phục,  $\varphi$  là hàm kích hoạt của bộ giải mã, và  $\mathbf{Z}$  là đầu ra của bộ mã hóa.

### 1.2.3 Mạng nơ-ron đồ thị không-thời gian

Mạng nơ-ron đồ thị không-thời gian (Spatial-Temporal Graph Neural Network - STGNN) [22] được đề xuất để mô hình hóa tính linh động và tính phụ thuộc lẫn nhau. Để thu được các tương quan không-thời gian của dữ liệu, các trạng thái ẩn của STGNN được truyền qua một mạng nơ-ron hồi quy (Recurrent Neural Network - RNN) dựa trên tích chập đồ thị. Đầu ra của STGNN tại time step ( $t$ ) là:

$$\mathbf{H}^{(t)} = \sigma(\mathbf{W}\mathbf{X}^{n(t)} + \mathbf{H}^{(t-1)} + b), \quad (1.4)$$

trong đó  $\mathbf{X}^{n(t)}$  là ma trận đặc trưng nút tại time step  $t$  và  $\mathbf{U} \in \mathbb{R}^{n \times n}$  là ma trận véc-tơ riêng với  $\mathbf{U}\mathbf{U}^T = \mathbf{I}$ .

### 1.2.4 Các kiến trúc mạng nơ-ron đồ thị khác

Ngoài ba kiến trúc GNN trên, một số mô hình GNN phổ biến khác sẽ được trình bày chi tiết trong Chương 2, trong đó nổi bật là Mạng nơ-ron tích chập đồ thị (Graph Convolutional Network - GCN) và Mạng đồ thị cơ chế chú ý (Graph Attention Network - GAT). GCN [23] được sử dụng trong bộ mã hóa của GAE để tính  $\mathbf{Z}$ :

$$\mathbf{Z} = \text{GCN}(\mathbf{H}^{(l)}, \mathbf{A}); \quad (1.5)$$

và trong STGNN để tính đầu ra tại time step  $t$ :

$$\mathbf{H}^{(t)} = \sigma(\text{GCN}(\mathbf{X}^{n(t)}, \mathbf{A}; \mathbf{W}) + \text{GCN}(\mathbf{H}^{(t-1)}, \mathbf{A}; \mathbf{U}) + b). \quad (1.6)$$

GAT [24] gỡ bỏ các lớp mạng kết nối đầy đủ (fully connected) trung gian và thay thế bằng cơ chế chú ý [25] mà vẫn duy trì cấu trúc đồ thị. Cơ chế attention cho phép mô hình học tính cục bộ thích ứng và linh động của các vùng lân cận để có thể dự đoán chính xác hơn. Cơ chế này sẽ được trình bày cụ thể ở Mục 1.3.

### **1.3 Cơ chế chú ý**

Trước khi các mô hình mạng nơ-ron được huấn luyện để có thể phát hiện tin giả, việc đầu tiên cần làm là xử lí các bản tin để chuyển nội dung bản tin thành các véc-tơ nhiều chiều mà mô hình có thể xử lí được. Để làm điều này, ta cần các mô hình xử lí ngôn ngữ tự nhiên mà trong đó cơ chế chú ý đóng vai trò quan trọng trong việc giúp mô hình hiểu chính xác ý nghĩa của câu văn. Cơ chế này còn được sử dụng để xây dựng mô hình GAT sẽ được trình bày ở Mục 2.2. Mục 1.3 sẽ trình bày cơ sở lí thuyết của cơ chế chú ý.

#### **1.3.1 Giới thiệu**

Cơ chế chú ý [25] ( cơ chế attention) là một bước đột phá trong nghiên cứu học sâu ở thập kỉ 2010s và là tiền đề khai sinh ra các kiến trúc mô hình hiện đại trong lĩnh vực xử lí ngôn ngữ tự nhiên (Natural Language Processing - NLP) như Transformer [26] hay BERT [27]. Cơ chế chú ý cải thiện các mô hình học sâu bằng cách tập trung có chọn lọc vào các phần quan trọng của đầu vào, từ đó cải thiện độ chính xác dự đoán và độ hiệu quả tính toán. Trong tâm lí học, sự tập trung (attention) là một quá trình nhận thức của việc tập trung có chọn lọc vào một hoặc một vài thứ trong khi bỏ qua các thứ khác. Mạng nơ-ron vốn được xây dựng để mô phỏng hoạt động não người, và cơ chế chú ý cũng là một nỗ lực để thực hiện cơ chế tập trung này của nhận thức con người.

Xét một ví dụ cơ chế tập trung có chọn lọc của não người. Giả sử ta có một bức ảnh tập thể và ta muốn đếm số người xuất hiện trong bức ảnh. Để làm điều này, ta chỉ cần đếm số đầu người và không cần quan tâm đến các thứ khác trong bức ảnh. Với một tác vụ khác như tìm kiếm một ai đó, não ta cũng sẽ tự biết chính xác cần phải làm gì, cụ thể là tìm kiếm dựa vào các đặc trưng của người đó trong khi bỏ qua các đặc trưng khác. Đây chính là sự tập trung mà não người thực hiện rất tốt.

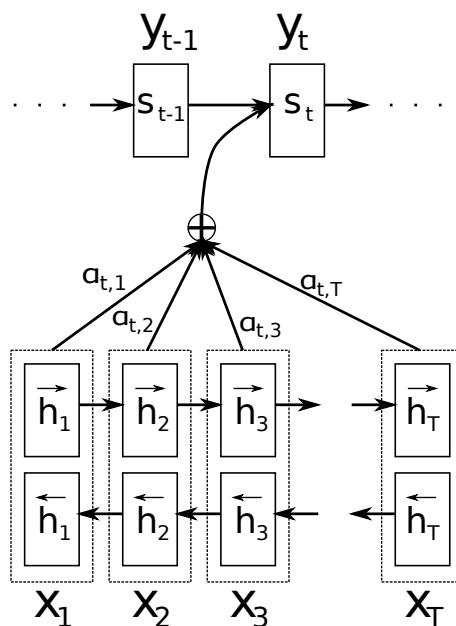
#### **1.3.2 Cơ chế chú ý trong học sâu**

Cơ chế chú ý nổi lên như là một sự cải tiến của hệ thống dịch máy bộ mã hoá - bộ giải mã (encoder-decoder) trong xử lí ngôn ngữ tự nhiên. Về sau, cơ chế này và các biến thể cũng được sử dụng trong các lĩnh vực khác, bao gồm thị giác máy tính. Trước khi cơ chế chú ý ra đời, các mô hình dịch máy được dựa trên cặp bộ mã hóa (encoder) - bộ giải mã (decoder), vốn là các lớp

RNN. Nó hoạt động theo hai bước sau:

1. Bộ mã hóa chuyển câu đầu vào thành một véc-tơ ngữ cảnh độ dài cố định (fixed-length) và là trạng thái ẩn (hidden state) cuối cùng của RNN. Đây được mong đợi là một bản tổng kết tốt của câu đầu vào. Các trạng thái trung gian khác bị bỏ qua, và trạng thái cuối sẽ là trạng thái ẩn khởi tạo của bộ giải mã.
2. Bộ giải mã dùng véc-tơ ngữ cảnh để sinh câu đầu ra lần lượt từng từ một.

Hạn chế chính của phương pháp này là nếu bộ mã hóa tạo ra một bản tổng kết kém, bản dịch cũng sẽ kém theo. Và thực tế đó đã xảy ra với các câu dài khi mà mô hình RNN (hay cả LSTM và GRU) không thể nhớ được chuỗi dài do véc-tơ ngữ cảnh có độ dài cố định. Cơ chế chú ý đã giải quyết vấn đề này bằng cách cho mô hình tập trung vào từng phần nhất định của dữ liệu thông qua việc gán trọng số. Bằng cách gán một trọng số tập trung cho mỗi từ trong một câu trong tương quan với các từ khác, ta có thể bảo toàn ngữ cảnh của các từ ngay cả với câu dài. Cơ chế chú ý được triển khai bằng cách dùng một mạng nơ-ron để học các phần quan trọng của dữ liệu và gán cho chúng trọng số lớn hơn khi đưa ra dự đoán.



**Hình 1.3:** Cơ chế chú ý [25].

Hình 1.3 minh họa cơ chế chú ý. Xét bài toán dịch máy có  $\mathbf{x} = (x_1, \dots, x_n)$  là câu đầu vào độ dài  $n$  và  $\mathbf{y} = (y_1, \dots, y_m)$  là câu đầu ra độ dài  $m$ . Một mô hình chuỗi hai chiều gồm hai trạng thái ẩn xuôi và ngược sẽ được sử dụng. Hai trạng thái ẩn này sẽ được ghép lại để biểu diễn trạng thái bộ mã hóa. Như

vậy, cả các từ phía trước và sau đều được dùng để tính độ tập trung vào một từ bất kì.

$$\mathbf{h}_i = \left[ \overrightarrow{\mathbf{h}}_i^\top; \overleftarrow{\mathbf{h}}_i^\top \right]^\top, \quad i = 1, \dots, n. \quad (1.7)$$

Trạng thái ẩn của bộ giải mã tại time step  $t$  là:

$$\mathbf{s}_t = f(\mathbf{s}_{t-1}, y_{t-1}, \mathbf{c}_t), \quad (1.8)$$

trong đó  $\mathbf{c}_t$  là véc-tơ ngữ cảnh của đầu ra  $y_t$  và bằng tổng có trọng số của các trạng thái ẩn  $\mathbf{h}_i$ :

$$\mathbf{c}_t = \sum_{i=1}^n \alpha_{t,i} \mathbf{h}_i, \quad (1.9)$$

trong đó  $\alpha_{t,i}$  được gọi là điểm căn chỉnh (alignment score) của cặp đầu vào  $x_i$  và đầu ra tại  $y_t$  dựa theo sự liên quan. Tập các trọng số  $\{\alpha_{t,i}\}$  thể hiện mức độ mà mỗi trạng thái ẩn nên tập trung vào mỗi trạng thái đầu ra.

$$\alpha_{t,i} = \frac{\exp(\text{score}(\mathbf{s}_{t-1}, \mathbf{h}_i))}{\sum_{k=1}^n \exp(\text{score}(\mathbf{s}_{t-1}, \mathbf{h}_k))}, \quad (1.10)$$

trong đó  $\text{score}(\cdot, \cdot)$  được học bằng một mạng nơ-ron truyền thẳng. Ở đây ta dùng hàm softmax để thu được các điểm căn chỉnh là các xác suất có tổng bằng 1.

### 1.3.3 Các loại cơ chế chú ý

#### a, Cơ chế Self-attention

Cơ chế self-attention cho phép các đầu vào tương tác lẫn nhau, nghĩa là có tính điểm chú ý của các đầu vào khác trong tương quan với một đầu vào.

#### b, Cơ chế chú ý tích vô hướng

Cơ chế chú ý tích vô hướng (dot-product attention) tính các trọng số chú ý bằng tích vô hướng:

$$\text{score}(\mathbf{s}_t, \mathbf{h}_i) = \mathbf{s}_t^\top \mathbf{h}_i. \quad (1.11)$$

### c, Cơ chế chú ý tích vô hướng điều chỉnh

Cơ chế chú ý tích vô hướng điều chỉnh (scaled dot-product attention) là biến thể của dot-product attention chia tích vô hướng cho kích thước đầu vào:

$$\text{score}(\mathbf{s}_t, \mathbf{h}_i) = \frac{\mathbf{s}_t^\top \mathbf{h}_i}{n}. \quad (1.12)$$

### d, Cơ chế chú ý đa đầu

Cơ chế chú ý đa đầu (multi-head attention) là cơ chế dùng nhiều lớp attention một cách đồng thời. Các đầu ra độc lập từ các lớp attention sau đó được nối lại với nhau. Cơ chế này cho phép mô hình có nhiều kiểu tập trung vào các phần của chuỗi.

## 1.4 Mô hình Transformer

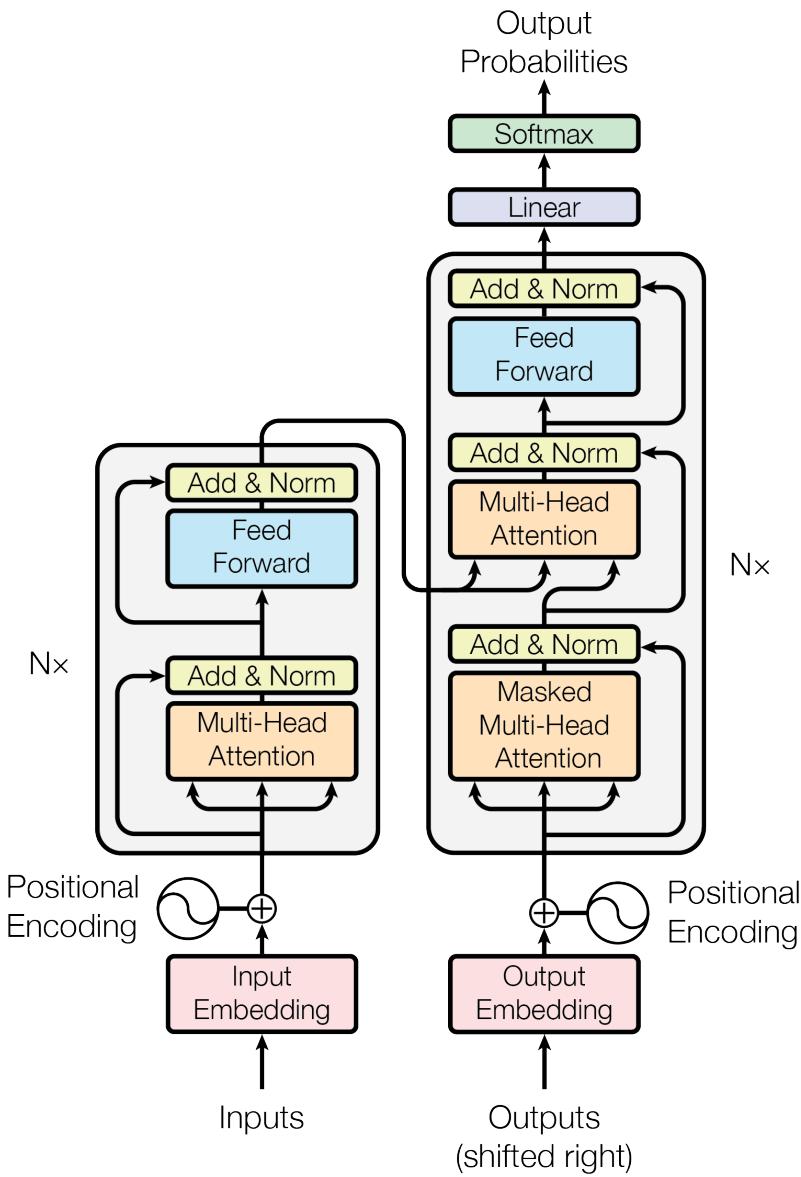
Trước khi đi vào một mô hình học biểu diễn dùng để sinh ra các véc-tơ đặc trưng từ nội dung bản tin là BERT ở Mục 1.5, ta cần biết đến kiến trúc Transformer. Kiến trúc này được xây dựng dựa trên cơ chế chú ý, có thể thực hiện tính toán song song ở một tầng đồng thời nắm bắt được các thông tin cách xa nhau trong chuỗi câu, giúp mô hình hiểu chính xác ý nghĩa câu với thời gian huấn luyện ngắn.

### 1.4.1 Giới thiệu

Trong bài toán dịch máy, mô hình [26] được xây dựng chỉ dựa trên cơ chế chú ý và thể hiện chất lượng vượt trội đồng thời yêu cầu ít thời gian huấn luyện hơn so với các mô hình truyền thống. Hình 1.4 thể hiện kiến trúc Transformer. Transformer vẫn đi theo kiến trúc cặp bộ mã hóa - bộ giải mã phổ biến của các mô hình dịch máy. Bộ mã hóa sẽ đưa chuỗi đầu vào  $\mathbf{x} = (x_1, \dots, x_n)$  thành một chuỗi biểu diễn liên tục  $\mathbf{z} = (z_1, \dots, z_n)$ . Từ  $\mathbf{z}$ , bộ giải mã sẽ sinh ra chuỗi đầu ra  $\mathbf{y} = (y_1, \dots, y_m)$  từng từ mỗi. Transformer sử dụng các lớp self-attention và lớp kết nối đầy đủ, theo điểm (point-wise, fully connected layer) xếp chồng lên nhau cho cả bộ mã hóa và bộ giải mã, lần lượt ở bên trái và bên phải Hình 1.4.

### 1.4.2 Bộ mã hóa

Bộ mã hóa được tạo nên từ 6 lớp xếp chồng giống nhau, mỗi lớp có hai lớp con. Lớp con thứ nhất là multi-head self-attention, còn lớp con thứ hai là một mạng nơ-ron truyền thẳng kết nối đầy đủ theo vị trí. Ngoài ra, một kết nối dư



**Hình 1.4:** Kiến trúc Transformer [26].

(residual connection) được thêm vào mỗi lớp con, sau đó bởi lớp chuẩn hóa (layer normalization). Đầu ra mỗi lớp con là  $\text{LayerNorm}(x + \text{Sublayer}(x))$ , trong đó  $\text{Sublayer}(x)$  là hàm của lớp con đó. Các lớp con và embedding đều có đầu ra kích thước  $d_{model} = 512$ .

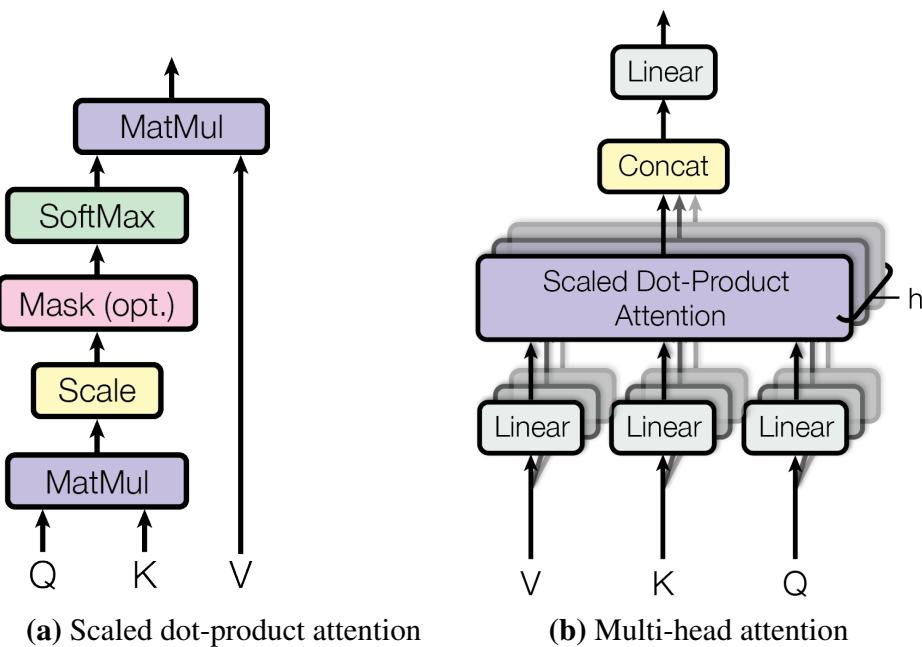
### 1.4.3 Bộ giải mã

Bộ giải mã cũng được tạo nên từ 6 lớp xếp chồng giống nhau. Ngoài hai lớp con như trong bộ mã hóa, bộ giải mã còn có thêm lớp con thứ ba là multi-head attention theo đầu ra của bộ mã hóa. Các kết nối dư cũng được áp dụng vào mỗi lớp con, sau đó là lớp chuẩn hóa. Lớp con self-attention còn được chỉnh sửa để tránh các vị trí tham gia vào các vị trí sau. Việc dùng mặt nạ (masking) này, cùng với việc các embedding đầu ra bị lệch một vị trí so với đầu vào, sẽ đảm bảo dự đoán tại vị trí  $i$  chỉ phụ thuộc vào các đầu ra đã

biết tại các vị trí nhỏ hơn  $i$ .

#### 1.4.4 Lớp chú ý

Một hàm chú ý có thể được mô tả là ánh xạ một query và một tập các cặp key-value vào một đầu ra, trong đó query, keys, values là các véc-tors. Đầu ra là một tổng có trọng số của các values, trong đó trọng số mỗi value được tính bằng một hàm tương thích của query với key tương ứng.



**Hình 1.5:** Cơ chế chú ý trong Transformer [26].

#### a, Cơ chế scaled dot-product attention trong Transformer

Cơ chế chú ý được sử dụng trong Transformer là scaled dot-product attention (Hình 1.5a). Đầu vào bao gồm các queries và keys có kích thước  $d_k$ , và các values có kích thước  $d_v$ . Ta tính tích vô hướng giữa query với tất cả các keys, chia cho  $\sqrt{d_k}$  và áp dụng một hàm softmax để thu được các trọng số trên các values.

Trên thực tế, ta tính sự chú đồng thời trên một tập các câu truy vấn trong một ma trận  $Q$ . Các từ khoá (keys) và giá trị (values) cũng hợp lại thành ma trận  $K$  và  $V$ . Ma trận đầu ra là:

$$\text{Attention}(Q, K, V) = \text{softmax} \left( \frac{QK^\top}{\sqrt{d_k}} \right) V. \quad (1.13)$$

## b, Cơ chế chú ý đa đầu trong Transformer

Thay vì dùng một hàm chú ý đơn lẻ, việc chiết tuyến tính các truy vấn, từ khoá và giá trị  $h$  lần với các phép chiết khác nhau xuông các kích thước  $d_k$  và  $d_v$  đã được chứng tỏ có hiệu quả hơn [26]. Trong mỗi lần chiết các truy vấn, từ khoá và giá trị, các hàm chú ý được thực hiện song song, tạo ra các véc-tơ đầu ra  $d_v$  chiết. Chúng được ghép lại với nhau và chiết một lần nữa, tạo ra đầu ra cuối cùng (Hình 1.5b).

Cơ chế chú ý đa đầu cho phép mô hình học đồng thời các thông tin từ các không gian con biểu diễn khác nhau tại các vị trí khác nhau.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h) W^O, \\ \text{head}_i = \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right), \quad (1.14)$$

trong đó các ma trận chiết là  $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$ ,  $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$ ,  $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$  và  $W_i^O \in \mathbb{R}^{hd_v \times d_{model}}$ .

Transformer dùng  $h = 8$  lớp chú ý song song,  $d_k = d_v = d_{model}/h = 64$ . Do các kích thước mỗi đầu bị giảm nên tổng chi phí tính toán là tương tự với việc dùng một lớp chú ý với kích thước đầy đủ.

**So sánh các truy vấn, từ khoá, giá trị với định nghĩa chú ý ở Mục 1.3:** Các khái niệm truy vấn, từ khoá, giá trị bắt nguồn từ bài toán truy xuất thông tin. Ví dụ, khi ta tìm kiếm một video trên YouTube, thì hệ thống so sánh văn bản ta gõ vào ô tìm kiếm (**truy vấn**) với các thông tin video (**từ khoá**, ví dụ tiêu đề, mô tả) trong cơ sở dữ liệu và trả về các videos phù hợp nhất (**giá trị**). So với định nghĩa chú ý ở Mục 1.3, truy vấn là tương ứng với trạng thái ẩn của bộ giải mã  $s_{t-1}$  tại time step  $t - 1$  còn từ khoá và giá trị là một và tương ứng với trạng thái ẩn của bộ mã hóa  $h_i$ .

### 1.4.5 Mạng nơ-ron truyền thẳng theo vị trí

Ngoài các lớp chú ý con, mỗi lớp của bộ mã hóa và bộ giải mã trong Transformer còn có một mạng truyền thẳng kết nối đầy đủ, được áp dụng vào từng vị trí một cách riêng rẽ và giống nhau. Mạng này bao gồm hai biến đổi tuyến tính với một hàm kích hoạt ReLU ở giữa:

$$\text{FFN}(x) = \max(0, xW_1 + b_1) W_2 + b_2. \quad (1.15)$$

Kích thước đầu vào và đầu ra của mạng là  $d_{model} = 512$ , và lớp bên trong có kích thước  $d_{ff} = 2048$ .

#### 1.4.6 Mã hóa theo vị trí

Vì Transformer không dùng các lớp hồi quy và tích chập, nên ta cần đưa thêm thông tin về vị trí các token trong chuỗi đầu vào. Để làm điều này, ta cộng thêm véc-tơ mã hóa vị trí (positional encoding) vào các embeddings đầu vào ở dưới các lớp bộ mã hóa và bộ giải mã. Véc-tơ mã hóa vị trí có cùng kích thước  $d_{model}$  như các embeddings. Transformer dùng các hàm sin và cosin để tính mã hóa theo vị trí:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right), \quad (1.16)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right), \quad (1.17)$$

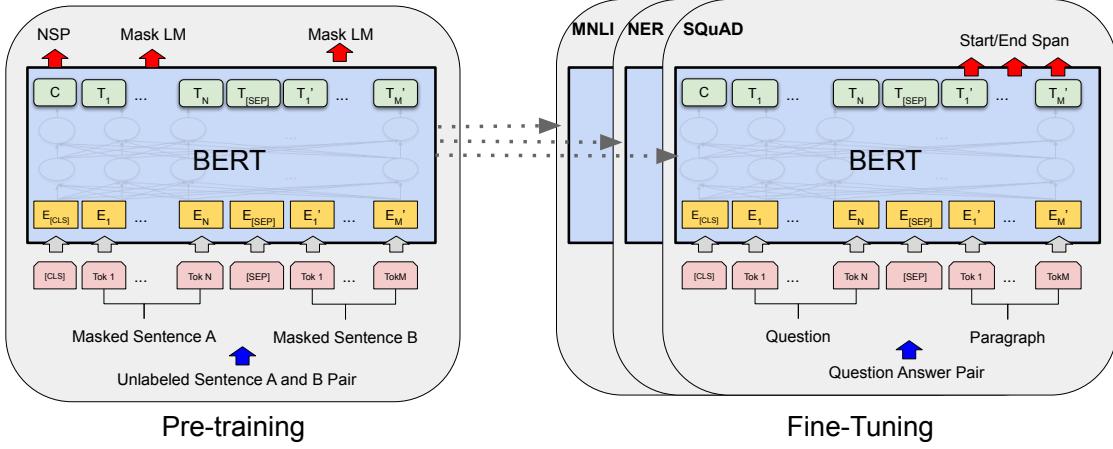
trong đó  $pos$  là vị trí của token và  $i$  là chỉ số phần tử của véc-tơ mã hóa vị trí.

### 1.5 Biểu diễn mã hóa hai chiều từ Transformers

Để sinh ra các véc-tơ embedding đặc trưng thể hiện nội dung bản tin, ta cần dùng đến một mô hình ngôn ngữ hiện đại là BERT. Đây là một mô hình biểu diễn từ theo hai chiều bằng kỹ thuật Transformer. BERT được thiết kế để huấn luyện trước các biểu diễn từ (pre-train word embedding). Cơ chế chú ý của Transformer sẽ truyền toàn bộ các từ trong câu vào mô hình đồng thời mà không cần quan tâm đến chiều của câu, cho phép mô hình học được ngữ cảnh của từ dựa trên toàn bộ các từ xung quanh nó.

#### 1.5.1 Giới thiệu

Biểu diễn mã hóa hai chiều từ Transformers (Bidirectional Encoder Representations from Transformers - BERT) [27] là một mô hình được tiền huấn luyện để học các biểu diễn hai chiều của văn bản, sau đó có thể được tinh chỉnh (finetune) mà chỉ cần thêm một lớp đầu ra cho các tác vụ. Hệ thống BERT gồm hai bước: tiền huấn luyện và tinh chỉnh (Hình 1.6). Đầu tiên, BERT được huấn luyện với dữ liệu không nhãn trên các tác vụ khác nhau. Tiếp theo, BERT được tinh chỉnh với dữ liệu có nhãn từ các tác vụ phía sau (downstream task). Một điểm nổi bật của BERT là kiến trúc thống nhất giữa các tác vụ khác nhau. Có rất ít sự khác nhau giữa kiến trúc đã được tiền huấn luyện và kiến trúc cho tác vụ phía sau.



**Hình 1.6:** Quá trình tiền huấn luyện và tinh chỉnh BERT [27].

## 1.5.2 Kiến trúc BERT

### a, Giới thiệu

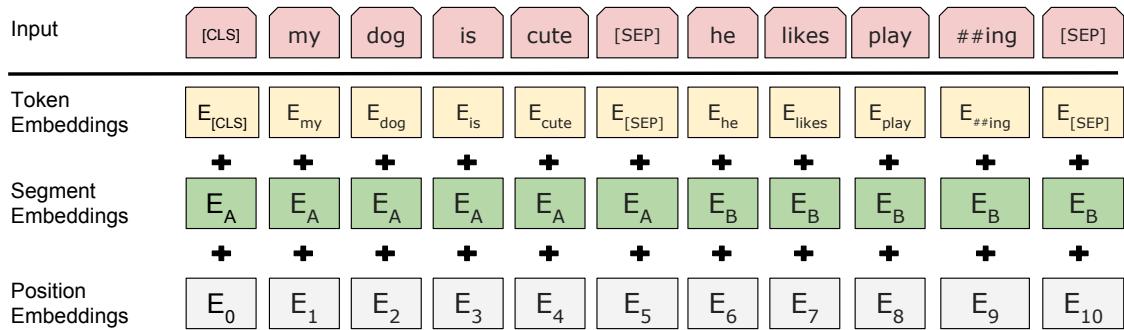
Kiến trúc BERT là một bộ mã hóa Transformer hai chiều nhiều lớp. Để BERT có thể xử lý các loại tác vụ phía sau khác nhau, biểu diễn đầu vào của BERT có thể biểu diễn cả một câu đơn và một cặp câu (ví dụ dạng <câu hỏi, câu trả lời>) trong một chuỗi token . Một câu ở đây có thể là một đoạn văn bản thay vì một câu trong ngôn ngữ thực tế. Một chuỗi ở đây có thể là một câu đơn hoặc một cặp hai câu.

BERT sử dụng WordPiece embeddings [28] với 30,000 từ vựng token. Token đầu tiên của mọi chuỗi luôn là một token phân loại đặc biệt (special classification token -  $[CLS]$ ). Trạng thái ẩn cuối tương ứng với token này được dùng như là biểu diễn chuỗi tổng hợp cho các tác vụ phân loại. Các cặp câu được gói lại thành một chuỗi đơn. Các câu được phân biệt theo hai cách. Thứ nhất, chúng được tách ra bằng một token đặc biệt (separate -  $[SEP]$ ). Thứ hai, một embedding đã được học sẽ được thêm vào mọi token để chỉ nó thuộc câu A hay câu B. Như trong Hình 1.6, embedding đầu vào là  $E$ , véc-tơ ẩn cuối cùng của token  $[CLS]$  là  $C \in \mathbb{R}^H$ , véc-tơ ẩn cuối cùng của token đầu vào thứ  $i$  là  $T_i \in \mathbb{R}^H$ , trong đó  $H$  là kích thước ẩn.

Với một token, biểu diễn đầu vào của nó được xây dựng bằng cách cộng các embeddings token, phân đoạn và vị trí tương ứng (Hình 1.7).

## 1.5.3 Tiền huấn luyện BERT

BERT được tiền huấn luyện bằng hai tác vụ không giám sát được trình bày trong mục này.



**Hình 1.7:** Biểu diễn đầu vào BERT [27].

### a, Tác vụ #1: Masked LM

Để huấn luyện một biểu diễn hai chiều, ta đặt mặt nạ (mask) một phần các tokens đầu vào một cách ngẫu nhiên, sau đó dự đoán các tokens bị ẩn này. Tác vụ này được gọi là "masked LM" (MLM). Trong trường hợp này, các véc-tơ ẩn cuối tương ứng với mask token được đưa vào một hàm softmax đầu ra trên tập từ vựng. BERT đặt mặt nạ cho 15% các WordPiece tokens trong mỗi chuỗi một cách ngẫu nhiên. Ở đây ta chỉ cần dự đoán các từ bị ẩn thay vì toàn bộ đầu vào.

### b, Tác vụ #2: Next sentence prediction (NSP)

Rất nhiều tác vụ phía sau như trả lời câu hỏi (Question Answering - QA) và suy diễn ngôn ngữ tự nhiên (Natural Language Inference - NLI) được dựa trên việc hiểu quan hệ giữa hai câu. Do đó, BERT được tiền huấn luyện cho tác vụ dự đoán câu tiếp theo nhị phân (next sentence prediction - NSP). Cụ thể, khi chọn các câu A và B cho mỗi dữ liệu huấn luyện, 50% số lần B là câu tiếp theo sau A, và 50% đó là một câu ngẫu nhiên từ kho ngữ liệu.

### c, Dữ liệu tiền huấn luyện

Đối với kho ngữ liệu tiền huấn luyện, BERT dùng BookCorpus (800 triệu từ) [29] và English Wikipedia (2.5 tỉ từ). BERT chỉ dùng đoạn văn bản và bỏ qua các danh sách, bảng và tiêu đề trong Wikipedia.

#### 1.5.4 Tính chỉnh BERT

Với các ứng dụng liên quan đến cặp văn bản, một khuôn mẫu thông thường là mã hóa độc lập các cặp văn bản trước khi áp dụng cơ chế chú ý chéo hai chiều (bidirectional cross attention). Thay vì thế, BERT dùng cơ chế self-attention để kết hợp hai giai đoạn này, vì mã hóa một cặp văn bản được ghép lại với nhau với self-attention sẽ bao gồm cả cơ chế chú ý chéo hai chiều giữa

hai câu.

Với mỗi tác vụ, ta đưa các đầu vào và đầu ra cụ thể vào BERT và tinh chỉnh tất cả tham số. Tại đầu vào, câu A và B từ tiền huấn luyện là tương tự với: 1) cặp câu trong diễn giải (paraphrasing); 2) cặp giả thuyết-tiền đề trong tác vụ kế thừa (entailment); 3) cặp câu hỏi-đoạn văn trong trả lời câu hỏi; và 4) cặp văn bản- suy biến trong phân loại văn bản hoặc gán thẻ chuỗi. Tại đầu ra, các biểu diễn token được đưa vào một lớp đầu ra cho tác vụ mức token, ví dụ như gán thẻ chuỗi hoặc trả lời câu hỏi, và biểu diễn [CLS] được đưa vào một lớp đầu ra cho tác vụ phân loại, ví dụ như tác vụ kế thừa hoặc phân tích cảm xúc.

## CHƯƠNG 2. CÁC MÔ HÌNH MẠNG NÓ-RON ĐỒ THỊ TRONG BÀI TOÁN XÁC THỰC THÔNG TIN

### 2.1 Mạng tích chập đồ thị

Mạng tích chập đồ thị (Graph Convolutional Network - GCN) [23] là một mô hình ổn định được áp dụng trên dữ liệu có cấu trúc đồ thị dựa trên một biến thể của CNN hoạt động trực tiếp trên đồ thị. GCN có kích thước tăng tuyến tính theo số lượng cạnh đồ thị và học các biểu diễn lớp ẩn mã hóa cả cấu trúc đồ thị cục bộ và đặc trưng các nút. GCN có thời gian huấn luyện ngắn hơn và độ chính xác cao hơn so với các mô hình khác vào thời điểm đó.

Trước khi xây dựng GCN, ta quay lại Phương trình 1.2 thể hiện sự lan truyền các lớp mạng GNN. Xét một luật lan truyền đơn giản:

$$f(\mathbf{H}^{(l)}, \mathbf{A}) = \sigma(\mathbf{A}\mathbf{H}^{(l)}\mathbf{W}^{(l)}), \quad (2.1)$$

trong đó  $\sigma$  là một hàm kích hoạt phi tuyến và  $\mathbf{W}^{(l)}$  là ma trận trọng số lớp  $l$ .

Mô hình này có hai hạn chế chính: việc nhân với ma trận  $\mathbf{A}$  nghĩa là với mọi nút, ta cộng các véc-tơ đặc trưng của tất cả các nút hàng xóm lại ngoài trừ nút đó, trừ khi có vòng khuyên (self-loop) trong đồ thị. GCN khắc phục điều này bằng cách thực thi các vòng khuyên trong đồ thị: đơn giản là cộng thêm ma trận đơn vị vào  $\mathbf{A}$ .

Hạn chế thứ hai là  $\mathbf{A}$  không được chuẩn hóa và việc nhân với  $\mathbf{A}$  sẽ thay đổi hoàn toàn khoảng giá trị của các véc-tơ đặc trưng. Chuẩn hóa  $\mathbf{A}$  sao cho tổng các hàng bằng một, nghĩa là  $\mathbf{D}^{-1}\mathbf{A}$ , trong đó  $\mathbf{D} \in \mathbb{R}^{N \times N}$  là ma trận bậc nút dạng đường chéo, có các phần tử  $D_{ii}$  là số nút hàng xóm của nút  $i$ . GCN dùng một chuẩn hóa đối xứng  $\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{1/2}$  để mô hình có tính linh hoạt hơn, vì đây không đơn thuần là trung bình của các nút hàng xóm. Cuối cùng ta có luật lan truyền của GCN là:

$$f(\mathbf{H}^{(l)}, \mathbf{A}) = \sigma(\hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}), \quad (2.2)$$

trong đó  $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$  và  $\hat{\mathbf{D}}$  là ma trận bậc nút dạng đường chéo của  $\hat{\mathbf{A}}$ .

## 2.2 Mạng đồ thị cơ chế chú ý

Mạng đồ thị cơ chế chú ý (Graph Attention Network - GAT) [24] là một kiến trúc mạng hoạt động trên dữ liệu có cấu trúc đồ thị, tận dụng các lớp masked self-attention để giải quyết các hạn chế trước đó của các phương pháp dựa trên tích chập đồ thị. Bằng cách xếp chồng các lớp trong đó các nút có thể tham gia vào các đặc trưng của hàng xóm, ta có thể chỉ định các trọng số cho các nút khác nhau trọng một lân cận, mà không cần các toán tử ma trận tốn chi phí (ví dụ nghịch đảo) hay phụ thuộc vào việc biết trước cấu trúc đồ thị.

Đầu tiên ta xét một lớp đồ thị cơ chế chú ý, vì đây là lớp duy nhất được dùng xuyên suốt kiến trúc GAT. Đầu vào của lớp là một tập các đặc trưng nút,  $\mathbf{h} = \{\vec{h}_1, \dots, \vec{h}_N\}$ ,  $\vec{h}_i \in \mathbb{R}^F$ , với  $F$  là số chiều đặc trưng mỗi nút. Lớp này tạo ra một tập mới các đặc trưng nút (có số chiều  $F'$ ),  $\mathbf{h}' = \{\vec{h}'_1, \dots, \vec{h}'_N\}$ ,  $\vec{h}'_i \in \mathbb{R}^{F'}$ .

Để có được năng lực biểu diễn đủ để chuyển các đặc trưng đầu vào thành các đặc trưng cấp cao, ta cần ít nhất một biến đổi tuyến tính có thể học được. Để làm điều này, một biến đổi tuyến tính chung dưới dạng một ma trận trọng số  $\mathbf{W} \in \mathbb{R}^{F' \times F}$  được áp dụng vào mọi nút. Sau đó ta thực hiện self-attention trên các nút bằng một cơ chế chú ý chung  $a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \rightarrow \mathbb{R}$  tính các hệ số chú ý:

$$e_{ij} = a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j), \quad (2.3)$$

xác định độ quan trọng của đặc trưng nút  $j$  đối với nút  $i$ . Trong công thức tổng quát, mô hình cho phép mọi nút tham gia vào các nút khác, bỏ qua thông tin cấu trúc đồ thị. Ta đưa cấu trúc đồ thị vào bằng cơ chế masked attention, trong đó ta chỉ tính  $e_{ij}$  cho các nút  $j \in \mathcal{N}_i$ , với  $\mathcal{N}_i$  là tập nút hàng xóm bậc nhất của  $i$  (bao gồm cả  $i$ ). Cuối cùng ta dùng hàm softmax để chuẩn hóa các hệ số:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}. \quad (2.4)$$

Cơ chế chú ý  $a$  là một mạng nơ-ron truyền thẳng một lớp biểu diễn bởi một véc-tơ trọng số  $\vec{a} \in \mathbb{R}^{2F'}$ , và áp dụng hàm LeakyReLU. Công thức đầy đủ

của các hệ số tính bởi cơ chế chú ý (Hình 2.1a) là:

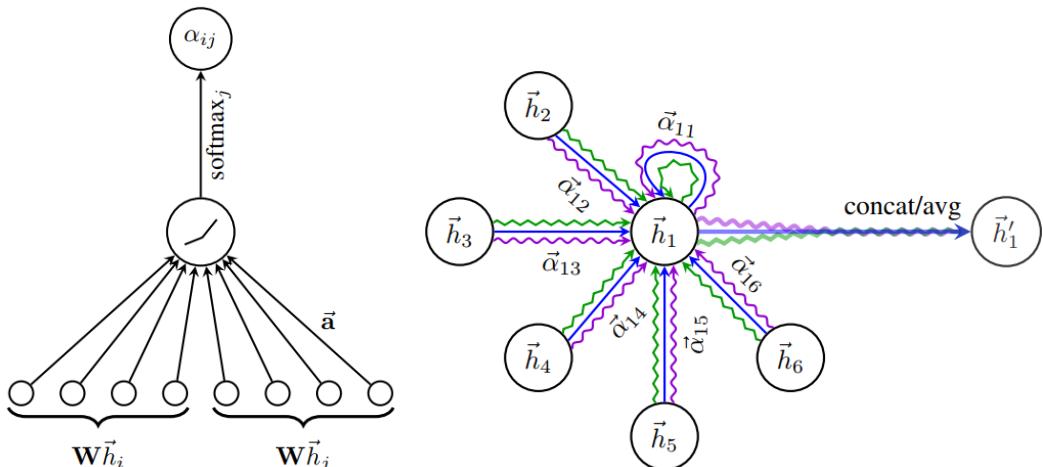
$$\alpha_{ij} = \frac{\exp \left( \text{LeakyReLU} \left( \vec{\mathbf{a}}^\top [\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j] \right) \right)}{\sum_{k \in \mathcal{N}_i} \exp \left( \text{LeakyReLU} \left( \vec{\mathbf{a}}^\top [\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k] \right) \right)}, \quad (2.5)$$

trong đó  $\|$  là toán tử ghép nối (concatenation).

Đặc trưng đầu ra cuối cùng là:

$$\vec{h}'_i = \sigma \left( \sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W}\vec{h}_j \right), \quad (2.6)$$

trong đó  $\sigma$  là một hàm phi tuyến.



(a) Cơ chế chú ý đơn trong GAT. (b) Multi-head attention trong GAT với  $K = 3$ .

**Hình 2.1:** Hai cơ chế chú ý trong GAT [24].

Để ổn định quá trình học của self-attention, ta mở rộng sang cơ chế multi-head attention (Hình 2.1b). Cụ thể,  $K$  cơ chế attention độc lập được áp dụng và các đặc trưng của chúng được ghép nối lại, tạo biểu diễn đặc trưng đầu ra:

$$\vec{h}'_i = \left\| \sigma \left( \sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j \right) \right\|, \quad (2.7)$$

trong đó  $\alpha_{ij}^k$  là hệ số chú ý chuẩn hóa, được tính bởi cơ chế chú ý thứ  $k$ ,  $\mathbf{W}^k$  là ma trận trọng số tương ứng. Đầu ra cuối cùng  $\mathbf{h}'$  có  $KF'$  đặc trưng cho mỗi nút.

Ở lớp cuối, ghép nối không còn ý nghĩa, thay vào đó ta lấy trung bình:

$$\vec{h}'_i = \sigma \left( \frac{1}{K} \sum_{k=1}^K \sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j \right). \quad (2.8)$$

## 2.3 Mô hình GraphSAGE

Các phương pháp hiện tại để xử lí embeddings của nút được dựa trên phân rã ma trận (matrix factorization) và yêu cầu tất cả các nút trong đồ thị hiện diện trong quá trình huấn luyện embeddings; các phương pháp này được gọi là *truyền dẫn* (*transductive*) và không tổng quát hóa các nút chưa biết. GraphSAGE (SAmple and aggreGatE) [30] là một framework *quy nạp* (*inductive*) tổng quát có thể tận dụng các thông tin đặc trưng nút để sinh ra các embeddings nút cho dữ liệu chưa biết. Ý tưởng chính của GraphSAGE là học cách tổng hợp thông tin từ vùng lân cận của mỗi nút (ví dụ, bậc và thuộc tinh văn bản của các nút liền kề).

### 2.3.1 Thuật toán sinh embedding

Giả sử mô hình đã được huấn luyện và các tham số được cố định, Thuật toán 1 mô tả quá trình sinh embedding (lan truyền thuận). Cụ thể, ta giả sử đã học được các tham số của  $K$  hàm tổng hợp  $\text{AGGREGATE}_k, \forall k \in \{1, \dots, K\}$  mà tổng hợp thông tin từ các nút hàng xóm, cũng như các ma trận trọng số  $\mathbf{W}^k, \forall k \in \{1, \dots, K\}$  mà được dùng để lan truyền thông tin giữa các lớp của mô hình. Tại mỗi vòng lặp, các nút tổng hợp thông tin từ hàng xóm của chúng, và qua các vòng lặp các nút có thêm ngày càng nhiều thông tin ở xa hơn trên đồ thị.

Trong Thuật toán 1, đầu vào là đồ thị  $\mathcal{G}(\mathcal{V}, \mathcal{E})$  và các đặc trưng nút  $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$ . Tại mỗi vòng lặp ngoài, mỗi nút  $v \in \mathcal{V}$  tổng hợp các biểu diễn của các nút liền kề,  $\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\}$ , vào một véc-tơ  $\mathbf{h}_{\mathcal{N}(v)}^k$ . Sau đó, ta ghép nối biểu diễn nút hiện tại  $\mathbf{h}_v^{k-1}$  với  $\mathbf{h}_{\mathcal{N}(v)}^k$ , và đưa véc-tơ được ghép nối vào một lớp kết nối đầy đủ với hàm kích hoạt  $\sigma$  phi tuyến, tạo ra biểu diễn  $\mathbf{h}_v^k$  để sử dụng ở vòng lặp kế tiếp. Để tiện lợi, ta ký hiệu đầu ra biểu diễn cuối cùng là  $\mathbf{z}_v \equiv \mathbf{h}_v^K, \forall v \in \mathcal{V}$ . Hàm tổng hợp AGGREGATE có nhiều kiến trúc khác nhau sẽ được trình bày ở Mục 2.3.3.

---

### Thuật toán 1: Thuật toán sinh embedding trong GraphSAGE

---

**Đầu vào:** Đồ thị  $\mathcal{G}(\mathcal{V}, \mathcal{E})$ ; các đặc trưng đầu vào  $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$ ; độ sâu  $K$ ; các ma trận trọng số  $\mathbf{W}^k, \forall k \in \{1, \dots, K\}$ ; hàm phi tuyến  $\sigma$ ; các hàm tổng hợp khả vi  $\text{AGGREGATE}_k, \forall k \in \{1, \dots, K\}$ ; hàm hàng xóm  $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

**Đầu ra :** Các biểu diễn véc-tơ  $\mathbf{z}_v$  for all  $v \in \mathcal{V}$

```
1  $\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V};$ 
2 for  $k = 1 \dots K$  do
3   for  $v \in \mathcal{V}$  do
4      $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k (\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\});$ 
5      $\mathbf{h}_v^k \leftarrow \sigma (\mathbf{W}^k \cdot \text{CONCAT} (\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k))$ 
6   end
7    $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{E}V$ 
8 end
9  $\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$ 
```

---

### 2.3.2 Học các tham số của GraphSAGE

Ta sử dụng một hàm măt măt dựa trên đồ thị trên các biểu diễn đầu ra  $\mathbf{z}_u, \forall u \in \mathcal{V}$  và cập nhật các ma trận trọng số  $\mathbf{W}^k, \forall k \in \{1, \dots, K\}$  và các tham số của các hàm tổng hợp. Hàm măt măt khiến các nút gần nhau có biểu diễn tương tự nhau, đồng thời khiến biểu diễn của các nút xa nhau trở nên rất khác biệt:

$$J_{\mathcal{G}}(\mathbf{z}_u) = -\log (\sigma(\mathbf{z}_u^\top \mathbf{z}_v)) - Q \cdot \mathbb{E}_{v_n \sim P_n(v)} \log (\sigma(-\mathbf{z}_u^\top \mathbf{z}_{v_n})), \quad (2.9)$$

trong đó  $v$  là một nút cùng xuất hiện gần  $u$  trên bước nhảy ngẫu nhiên có độ dài cố định,  $\sigma$  là hàm sigmoid,  $P_n$  là phân phối lấy mẫu âm, và  $Q$  là số lượng mẫu âm. Khác với các phương pháp trước đó, biểu diễn  $\mathbf{z}_u$  được sinh ra từ các đặc trưng trong vùng lân cận của một nút thay vì học một embedding riêng cho từng nút.

### 2.3.3 Các kiến trúc hàm tổng hợp

Các hàng xóm của một nút không có thứ tự nhất định, do đó các hàm tổng hợp phải hoạt động trên một tập các véc-tors không có thứ tự. Lý tưởng là, một hàm tổng hợp là đối xứng (bất biến theo các hoán vị đầu vào) trong khi

vẫn có thể huấn luyện được và duy trì khả năng biểu diễn tốt. Tính đối xứng của hàm tổng hợp đám bảo mạng nơ-ron có thể được huấn luyện và áp dụng vào các tập đặc trưng nút hàng xóm có thứ tự tùy ý. Ta xét ba hàm tổng hợp:

### a, Hàm tổng hợp trung bình

Hàm này lấy trung bình theo từng phần tử của các véc-tos trong  $\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\}$ . Hàm tổng hợp trung bình là gần tương tự với luật lan truyền tích chập trong framework GCN truyền dẫn [23]. Cụ thể, ta có được một biến thể quy nạp của GCN bằng cách thay dòng 4 và 5 trong Thuật toán 1 bằng:

$$\mathbf{h}_v^k \leftarrow \sigma(\mathbf{W} \cdot \text{MEAN} (\{\mathbf{h}_v^{k-1}\} \cup \{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})) . \quad (2.10)$$

### b, Hàm tổng hợp LSTM

Hàm này được dựa trên kiến trúc Long Short-Term Memory (LSTM) [31]. So với hàm tổng hợp trung bình, hàm tổng hợp LSTM có khả năng biểu diễn lớn hơn. Tuy nhiên, hàm này vốn không đối xứng, vì chúng xử lý đầu vào theo tuần tự. Ta dùng LSTM để hoạt động trên một tập không có thứ tự bằng cách áp dụng LSTM vào một hoán vị ngẫu nhiên các hàng xóm của một nút.

### c, Hàm tổng hợp Pooling

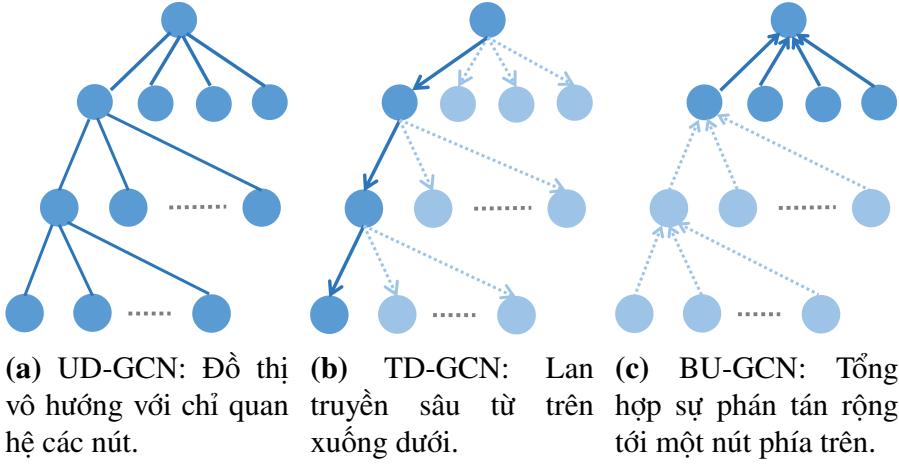
Hàm này vừa đối xứng vừa có thể được huấn luyện. Trong hàm này, mỗi véc-tor hàng xóm được đưa qua một mạng nơ-ron kết nối đầy đủ, rồi áp dụng một toán tử max-pooling theo từng phần tử để tổng hợp thông tin trên các tập hàng xóm:

$$\text{AGGREGATE}_k^{\text{pool}} = \max (\{\sigma (\mathbf{W}_{\text{pool}} \mathbf{h}_u^k + \mathbf{b}), \forall u \in \mathcal{N}(v)\}) , \quad (2.11)$$

## 2.4 Mạng tích chập đồ thị hai chiều

Mạng tích chập đồ thị hai chiều (Bi-Directional Graph Convolutional Network - BiGCN) [32] được đề xuất để xử lý sự lan truyền (propagation) và sự phát tán (dispersion) của tin tức bằng hai phần: 1) Mạng tích chập đồ thị từ trên xuống dưới (Top-Down GCN - TD-GCN) và 2) Mạng tích chập đồ thị từ dưới lên trên (Bottom-Up GCN - BU-GCN). Trên Hình 2.2a, GCN vô hướng (undirected GCN - UD-GCN) chỉ tổng hợp thông tin dựa trên các quan hệ giữa các tin tức liên quan nhưng không có thứ tự. Mặc dù UD-GCN có thể xử lý các đặc trưng cấu trúc toàn cục của sự lan truyền tin tức, nó không xét

đến hướng lan truyền. Trong khi đó, TD-GCN (Hình 2.2b) chuyển tiếp thông tin từ nút gốc tới các nút con để xây dựng sự lan truyền, và BU-GCN (Hình 2.2c) tổng hợp thông tin từ các nút con để biểu diễn sự phân tán. Các biểu diễn của sự lan truyền và sự phân tán từ embedding của TD-GCN và BU-GCN sẽ được kết hợp lại để tính kết quả cuối cùng. Ngoài ra, BiGCN còn ghép nối đặc trưng của nút gốc với các đặc trưng tại mỗi lớp GCN để tăng ảnh hưởng từ nút gốc, đồng thời áp dụng DropEdge [33] để tránh over-fitting.

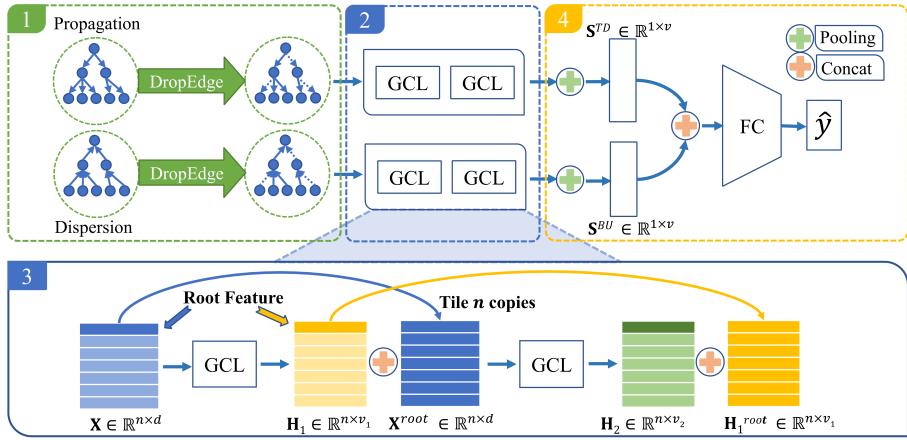


**Hình 2.2:** Các mô hình lan truyền trong GCN [32].

Gọi  $C = \{c_1, \dots, c_m\}$  là tập dữ liệu sự kiện tin tức với  $m$  là số lượng sự kiện;  $c_i = \{r_i, w_1^i, \dots, w_{n_i-1}^i, G_i\}$  với  $n_i$  là số lượng bài đăng trong sự kiện  $c_i$ ,  $r_i$  là bài đăng gốc,  $w_j^i$  là bài đăng phản hồi thứ  $j$ , và  $G_i$  là đồ thị lan truyền. Cụ thể,  $G_i = \langle V_i, E_i \rangle$  với  $r_i$  là nút gốc,  $V_i = \{r_i, w_1^i, \dots, w_{n_i-1}^i\}$ , và  $E_i = \{e_{st}^i \mid s, t = 0, \dots, n_i - 1\}$  là tập các cạnh từ bài đăng được phản hồi tới bài đăng được chia sẻ hoặc bài đăng phản hồi. Ví dụ, nếu  $w_2^i$  có một phản hồi tới  $w_1^i$ , ta có một cạnh có hướng  $w_1^i \rightarrow w_2^i$  ( $e_{12}^i$ ). Nếu  $w_1^i$  có một phản hồi tới  $r_i$ , ta có một cạnh có hướng  $r_i \rightarrow w_1^i$  ( $e_{01}^i$ ). Ý tưởng chính của BiGCN là học các biểu diễn bậc cao từ cả sự lan truyền và sự phân tán tin tức. Hình 2.3 thể hiện mô hình BiGCN gồm bốn bước.

#### 2.4.1 Xây dựng đồ thị lan truyền và phân tán

Dựa trên các quan hệ chia sẻ lại và trả lời bài đăng, ta xây dựng cấu trúc lan truyền  $\langle V, E \rangle$  cho một tin tức  $c_i$ . Đặt  $\mathbf{A} \in \mathbb{R}^{n_i \times n_i}$  và  $\mathbf{X}$  lần lượt là ma trận kề và ma trận đặc trưng của  $c_i$ .  $\mathbf{A}$  chỉ có các cạnh từ các nút trên xuống các nút dưới như ở Hình 2.2b. Gọi tổng số cạnh trong đồ thị của  $\mathbf{A}$  là  $N_e$  và gọi  $p$  là tỉ lệ số cạnh bị bỏ đi để tránh over-fitting. Tại mỗi epoch, ma trận kề



**Hình 2.3:** Mô hình BiGCN. GCL là lớp GCN (GCN layer) [32].

sau khi áp dụng DropEdge là:

$$\mathbf{A}' = \mathbf{A} - \mathbf{A}_{drop}, \quad (2.12)$$

trong đó  $\mathbf{A}_{drop}$  là ma trận tạo từ  $p \cdot N_e$  cạnh lấy ngẫu nhiên từ tập cạnh gốc.

TD-GCN và BU-GCN lần lượt có ma trận kè là  $\mathbf{A}^{TD} = \mathbf{A}'$  và  $\mathbf{A}^{TD} = \mathbf{A}'^\top$ . Cả TD-GCN và BU-GCN có cùng ma trận đặc trưng  $\mathbf{X}$ .

#### 2.4.2 Tính toán các biểu diễn nút bậc cao

Sau khi áp dụng DropEdge, ta sẽ có được các đặc trưng lan truyền từ trên xuống và từ dưới lên bằng TD-GCN và BU-GCN. Thay  $\mathbf{A}^{TD}$  và  $\mathbf{X}$  vào Phương trình 2.1, ta có các đặc trưng ẩn từ trên xuống của hai lớp TD-GCN:

$$\mathbf{H}_1^{TD} = \sigma \left( \hat{\mathbf{A}}^{TD} \mathbf{X} \mathbf{W}_0^{TD} \right), \quad (2.13)$$

$$\mathbf{H}_2^{TD} = \sigma \left( \hat{\mathbf{A}}^{TD} \mathbf{H}_1 \mathbf{W}_1^{TD} \right), \quad (2.14)$$

với  $\sigma$  là hàm ReLU, Dropout [34] được áp dụng ở các lớp GCN để tránh overfitting. Tương tự, ta cũng tính các đặc trưng ẩn từ dưới lên  $\mathbf{H}_1^{BU}$  và  $\mathbf{H}_2^{BU}$  cho BU-GCN.

#### 2.4.3 Tăng cường đặc trưng nút gốc

Bài đăng gốc luôn có nhiều thông tin để gây ảnh hưởng rộng, do đó ta cần sử dụng tốt hơn thông tin từ bài đăng gốc, và học các biểu diễn nút chính xác hơn từ quan hệ giữa các nút và bài đăng gốc. Bên cạnh TD-GCN và BU-GCN, ta thực hiện tăng cường đặc trưng gốc để cải thiện hiệu năng mô hình. Cụ thể, đối với TD-GCN tại lớp GCN thứ  $k$ , ta ghép nối các vec-tơ đặc trưng ẩn của mỗi nút với vec-tơ đặc trưng ẩn của nút gốc từ lớp GCN thứ  $k-1$  để

xây dựng ma trận đặc trưng mới là:

$$\tilde{\mathbf{H}}_k^{TD} = \mathbf{H}_k^{TD} \parallel (\mathbf{H}_{k-1}^{TD})^{root}, \quad (2.15)$$

với  $\mathbf{H}_0^{TD} = \mathbf{X}$ . Ta biểu diễn TD-GCN với tăng cường đặc trưng gốc bằng cách thay  $\mathbf{H}_1^{TD}$  trong Phương trình 2.14 bằng  $\tilde{\mathbf{H}}_1^{TD} = \mathbf{H}_1^{TD} \parallel \mathbf{X}^{root}$ , và tính  $\tilde{\mathbf{H}}_2^{TD}$  như sau:

$$\mathbf{H}_2^{TD} = \sigma \left( \hat{\mathbf{A}}^{TD} \tilde{\mathbf{H}}_1^{TD} \mathbf{W}_1^{TD} \right), \quad (2.16)$$

$$\tilde{\mathbf{H}}_2^{TD} = \mathbf{H}_2^{TD} \parallel (\mathbf{H}_1^{TD})^{root}. \quad (2.17)$$

Tương tự, ta cũng tính được các ma trận đặc trưng ẩn của BU-GCN với tăng cường đặc trưng gốc là  $\tilde{\mathbf{H}}_1^{BU}$  và  $\tilde{\mathbf{H}}_2^{BU}$ .

#### 2.4.4 Các biểu diễn sự lan truyền và sự phát tán để phân loại tin tức

Các biểu diễn sự lan truyền và sự phát tán lần lượt là các tổng hợp từ các biểu diễn nút của TD-GCN và BU-GCN. Ở đây ra dùng mean-pooling để tổng hợp thông tin từ hai tập biểu diễn nút này.

$$\mathbf{S}^{TD} = \text{MEAN} \left( \tilde{\mathbf{H}}_2^{TD} \right), \quad (2.18)$$

$$\mathbf{S}^{BU} = \text{MEAN} \left( \tilde{\mathbf{H}}_2^{BU} \right). \quad (2.19)$$

Sau đó ta ghép nối biểu diễn của sự lan truyền và sự phân tán để kết hợp thông tin:

$$\mathbf{S} = \mathbf{S}^{TD} \parallel \mathbf{S}^{BU}. \quad (2.20)$$

Cuối cùng, nhãn ý được tính bằng một lớp kết nối đầy đủ và hàm sigmoid:

$$\hat{\mathbf{y}} = \sigma(F\mathbf{C}(\mathbf{S})). \quad (2.21)$$

### 2.5 Mạng nơ-ron đồ thị với học liên tục

Mạng nơ-ron đồ thị với học liên tục (Graph Neural Network with Continual Learning - GNNCL) [9] được đề xuất nhằm khắc phục vấn đề GNN được huấn luyện trên một tập dữ liệu có hiệu năng kém trên dữ liệu mới, chưa được thấy. Học tiệm tiến (incremental learning) không thể giải quyết vấn đề này,

khi kết quả thí nghiệm cho thấy nếu tiếp tục huấn luyện mô hình trên tập dữ liệu mới thì nó sẽ chỉ thể hiện tốt trên dữ liệu mới và quên kiến thức đã học trên dữ liệu cũ. GNNCL khắc phục bằng hai phương pháp học liên tục:

- Bộ nhớ phân đoạn gradient (Gradient Epsiodic Memory - GEM) [35]: GEM sử dụng bộ nhớ phân đoạn để lưu một lượng các mẫu từ tác vụ trước, và khi học tác vụ mới  $t$ , nó không cho phép mất mát trên các mẫu được lưu này tăng lên so với khi việc học tác vụ  $t - 1$  kết thúc.
- Hợp nhất trọng số đàn hồi (Elastic Weight Consolidation - EWC) [36]: Hàm mất mát có một thành phần phạt (penalty term) bậc hai vào sự thay đổi tham số nhằm ngăn chặn cập nhật lớn vào các tham số quan trọng với tác vụ cũ.

Việc học trên hai tập dữ liệu  $\mathcal{D}_1$  và  $\mathcal{D}_2$  được xem là hai tác vụ. Ta huấn luyện mô hình với tác vụ đầu như thông thường; với tác vụ thứ hai, ta áp dụng GEM và EWC.

- Gọi  $\theta_1$  là tập tham số mô hình sau tác vụ đầu, và  $\mathcal{M}$  là tập các mẫu lấy từ tập dữ liệu đầu. Bài toán tối ưu của GEM là:

$$\begin{aligned} \min_{\theta} \sum_{(G_i, y_i) \in \mathcal{D}_2} & loss \left( f \left( A_i^{(k)}, H_i^{(k)}; \theta^{(k)} \right), y_i \right) \\ \text{với điều kiện } & \sum_{(G_j, y_j) \in \mathcal{M}} loss \left( f \left( A_j^{(k)}, H_j^{(k)}; \theta^{(k)} \right), y_j \right) \\ & \leq \sum_{(G_j, y_j) \in \mathcal{M}} loss \left( f \left( A_j^{(k)}, H_j^{(k)}; \theta_1^{(k)} \right), y_j \right) \end{aligned} \quad (2.22)$$

- Đặt  $\lambda$  là trọng số regularization,  $F$  là ma trận Fisher information, và  $\theta_{\mathcal{D}_1}^*$  là tập tham số phân phối Gaussian được dùng bởi EWC để xấp xỉ hậu nghiệm của  $p(\theta | \mathcal{D}_1)$ , hàm mất mát của EWC là:

$$\sum_{(G_i, y_i) \in \mathcal{D}_2} loss \left( f \left( A_i^{(k)}, H_i^{(k)}; \theta^{(k)} \right), y_i \right) + \frac{\lambda}{2} F (\theta - \theta_{\mathcal{D}_1}^*)^2. \quad (2.23)$$

## CHƯƠNG 3. THÍ NGHIỆM HỆ THỐNG PHÁT HIỆN TIN GIẢ UPFD VÀ KẾT QUẢ

### 3.1 Hệ thống phát hiện tin giả dựa trên sở thích người dùng UPFD

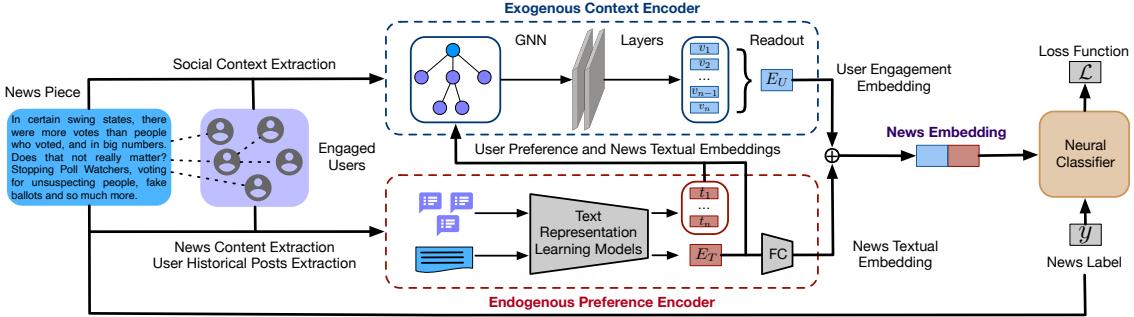
#### 3.1.1 Tổng quan

Phần lớn các thuật toán phát hiện tin giả tập trung vào khai thác nội dung tin tức và ngữ cảnh ngoại sinh (exogeneous) để tìm ra dấu hiệu lừa đảo, trong khi sở thích nội sinh (endogeneous) của người dùng khi họ quyết định chia sẻ tin giả hay không bị bỏ qua. Ở đây có thể hiểu ngữ cảnh ngoại sinh là tương tác giữa các người dùng với nhau và với các bản tin hay bài đăng trên mạng xã hội, còn sở thích nội sinh là xu hướng người dùng thích hoặc không thích một nội dung tin tức nào đó. Khả năng một người dùng chia sẻ tin giả phụ thuộc vào niềm tin, và lịch sử hoạt động trên mạng xã hội của họ sẽ cung cấp nhiều thông tin quý giá về sở thích người dùng. Hệ thống phát hiện tin giả User Preference-aware Fake News Detection (UPFD) [37] được xây dựng để mô hình hóa đồng thời ngữ cảnh ngoại sinh và nội sinh (Hình 3.1). Cụ thể, UPFD gồm ba thành phần chính:

1. Để mô hình hóa sở thích nội sinh của người dùng, ta mã hóa nội dung tin tức và các bài đăng cũ của người dùng bằng phương pháp học biểu diễn văn bản.
2. Để có được ngữ cảnh ngoại sinh, ta xây dựng một đồ thị lan truyền cho mỗi bản tin dựa trên lịch sử chia sẻ của nó. Bài đăng bản tin được gán là nút gốc, và các nút khác biểu diễn các người dùng chia sẻ bản tin.
3. Để kết hợp thông tin ngoại sinh và nội sinh, ta dùng các biểu diễn véc-tơ của bản tin và người dùng làm đặc trưng nút và dùng các GNNs để học một embedding tương tác chung của người dùng. Embedding tương tác của người dùng và embedding văn bản của bản tin sẽ được dùng để huấn luyện một bộ phân loại nơ-ron để phát hiện tin giả.

#### 3.1.2 Mã hóa sở thích nội sinh

Để mã hóa thông tin văn bản của bản tin và sở thích người dùng, ta dùng hai phương pháp học biểu diễn dựa trên tiền huấn luyện ngôn ngữ. Với các véc-tors word2vec [38], ta chọn 680,000 véc-tors 300 chiều được tiền huấn luyện bằng spaCy [39]. Ta cũng dùng các embeddings tiền huấn luyện BERT



**Hình 3.1:** Hệ thống phát hiện tin giả UPFD [37].

[27] để mã hóa nội dung bản tin và các bài đăng trong lịch sử. Tiếp theo, vì spaCy có 680,000 từ, ta lấy trung bình các véc-tos của từ hiện tại trong 200 bài đăng gần nhất để có được biểu diễn sở thích người dùng. Embedding văn bản của bản tin cũng được tính tương tự. Đối với BERT, nội dung bản tin được mã hóa với đồ dài chuỗi đầu vào tối đa (512 tokens). Vì độ dài chuỗi đầu vào của BERT bị giới hạn, ta không thể dùng BERT để mã hóa 200 bài đăng thành một chuỗi, do đó ta mã hóa mỗi bài đăng riêng biệt và lấy trung bình của chúng để tính biểu diễn sở thích một người dùng. Nhìn chung một bài đăng ngắn hơn bản tin, ta đặt độ dài chuỗi đầu vào tối đa của BERT là 16 tokens để mã hóa các bài đăng nhanh hơn.

### 3.1.3 Trích xuất ngữ cảnh ngoại sinh

Với một bản tin trên mạng xã hội, ngữ cảnh ngoại sinh của người dùng bao gồm tất cả người dùng tương tác với bản tin. Ta dùng thông tin chia sẻ bản tin để xây dựng đồ thị lan truyền tin tức. Ví dụ trên Hình 3.1, đó là một đồ thị dạng cây trong đó nút gốc là bản tin và các nút khác là các người dùng chia sẻ bản tin gốc. Ta định nghĩa một bản tin là  $v_1$ , và  $\{v_2, \dots, v_n\}$  là danh sách các người dùng chia sẻ  $v_1$  được sắp xếp theo thứ tự thời gian. Ta cũng định nghĩa hai luật dưới đây để xác định đường lan truyền tin tức:

- Với mọi tài khoản  $v_i$ , nếu  $v_i$  chia sẻ dùng một bản tin sau ít nhất một tài khoản trong  $\{v_2, \dots, v_n\}$ , ta ước lượng sự lan truyền từ tài khoản có mốc thời gian gần nhất với  $v_i$ . Vì các bài đăng mới nhất được hiện lên đầu tiên trên dòng thời gian của mạng xã hội, chúng có xác suất được chia sẻ lại cao hơn.
- Nếu tài khoản  $v_i$  không theo dõi tài khoản nào trong chuỗi chia sẻ, bao gồm cả tài khoản đăng bài gốc, ta ước lượng sự lan truyền từ tài khoản có

số lượng người theo dõi lớn nhất, vì các bài đăng từ tài khoản có nhiều người theo dõi có khả năng cao hơn được chia sẻ lại bởi các người dùng khác.

### 3.1.4 Tổng hợp thông tin

Đầu tiên, ta tổng hợp thông tin ngoại sinh và nội sinh bằng GNN. Với một GNN, embedding văn bản tin tức và embedding sở thích người dùng có thể được lấy làm đặc trưng nút. Với đồ thị lan truyền tin tức, GNN tổng hợp các đặc trưng từ các nút liền kề để học embedding của một nút. Ta cũng áp dụng một hàm đọc (readout function) trên các embeddings nút để lấy embedding đồ thị (nghĩa là embedding tương tác người dùng). Tiếp theo, vì nội dung bản tin thường có nhiều dấu hiệu về độ uy tín của bản tin, ta kết hợp embedding văn bản tin tức và và embedding tương tác người dùng bằng ghép nối (concatenation) thành embedding cuối cùng để làm giàu thông tin embedding bản tin. Embedding tin tức đã kết hợp cuối cùng được đưa qua một mạng nơ-ron có hai neurons đầu ra thể hiện xác suất tin thật và giả.

## 3.2 Hàm mất mát trong bài toán phát hiện tin giả

Bài toán phát hiện tin giả có thể được xem là bài toán phân loại nhị phân với hai nhãn  $y = 1$  (tin giả) và  $y = 0$  (tin thật). Gọi  $\mathbf{z}$  là logit đầu ra (giá trị sau hàm softmax ở lớp cuối cùng của mạng), tất cả các mô hình đều có hàm mất mát là:

$$\mathcal{L} = - \sum \log (\mathbf{z}_y), \quad (3.1)$$

trong đó  $\mathbf{z}_y \in [0, 1]$  là giá trị logit đầu ra (nghĩa là xác suất) tương ứng với nhãn đúng. Việc tối thiểu hàm này sẽ khiến giá trị xác suất sau hàm softmax tương ứng với nhãn đúng lớn lên, vì khi  $\mathbf{z}_y \rightarrow 1$  thì  $-\log (\mathbf{z}_y) \rightarrow 0$ ; ngoài ra khi  $\mathbf{z}_y \rightarrow 0$  thì  $-\log (\mathbf{z}_y) \rightarrow \infty$ , nên hàm mất mát này trừng phạt rất nặng các dự đoán sai.

## 3.3 Thiết lập thí nghiệm

### 3.3.1 Tập dữ liệu

Tập dữ liệu FakeNewsNet [40] được sử dụng vì nó bao gồm cả nội dung tin tức và thông tin tương tác xã hội của chúng trên Twitter. Tập dữ liệu này bao gồm thông tin tin tức thật và giả từ hai websites kiểm chứng sự thật là *politifact.com* và *gossipcop.com*, cùng với các tương tác xã hội liên quan trên Twitter, trong đó có lịch sử bài đăng chia sẻ của tất cả tài khoản được thu thập

bằng Twitter Developer API. Tập dữ liệu cung cấp lịch sử 299 bài đăng gần nhất trên Twitter của mỗi người dùng, tổng cộng có gần 20 triệu bài đăng. Ở đây cần lưu ý dữ liệu 20 triệu bài đăng này được dùng để mô hình hóa sở thích nội sinh của người dùng, không liên quan đến các bài đăng tin tức dùng để huấn luyện và đánh giá mô hình. Với các người dùng có tài khoản bị vô hiệu hóa, các bài đăng được lấy mẫu ngẫu nhiên từ các người dùng khác mà tương tác với cùng bản tin, thay vì xóa tài khoản không hoạt động vì làm thế sẽ phá vỡ cấu trúc lan truyền ban đầu và dẫn đến mã hóa ngữ cảnh ngoại sinh kém hiệu quả. Ngoài ra, các kí tự đặc biệt như hay đường dẫn URLs cũng được gỡ bỏ trước khi áp dụng các phương pháp học biểu diễn. Mỗi đồ thị trong tập dữ liệu là một đồ thị dạng cây với nút gốc là bản tin và các nút lá là các người dùng Twitter đã chia sẻ lại bản tin gốc. Hai nút người dùng có một cạnh nếu một người dùng chia sẻ lại bài đăng bản tin từ người dùng khác. Thống kê về tập dữ liệu được thể hiện trên Bảng 3.1.

**Bảng 3.1:** Thống kê về tập dữ liệu.

Tập dữ liệu	#Đồ thị (#Giả)	#Nút	#Cạnh	#TB nút / đồ thị
Politifact	314 (157)	41,054	40,740	131
Gossipcop	5,464 (2,732)	314,262	308,798	58

### 3.3.2 Chỉ số đánh giá

Trong bài toán phát hiện tin giả, hai chỉ số đánh giá được dùng là Accuracy (ACC) và F1 score (F1), được định nghĩa như sau:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}, \quad (3.2)$$

$$\text{Precision} = \frac{TP}{TP + FP}, \quad (3.3)$$

$$\text{Recall} = \frac{TP}{TP + FN}, \quad (3.4)$$

$$F1 = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}}, \quad (3.5)$$

trong đó TP, FP, TN, FN là các chỉ số đánh giá trong bài toán phân loại nhị phân gồm hai nhãn 1 (positive) và 0 (negative), được định nghĩa như sau:

- True Positive (TP): Số mẫu có dự đoán là 1 và nhãn đúng là 1.

- False Positive (FP): Số mẫu có dự đoán là 1 và nhãn đúng là 0.
- True Negative (TN): Số mẫu có dự đoán là 0 và nhãn đúng là 0.
- False Negative (FN): Số mẫu có dự đoán là 0 và nhãn đúng là 1.

Các khái niệm này được minh họa bằng một ma trận nhầm lẫn (confusion matrix) trên Bảng 3.2. Ngoài ra, trong công thức tính F1, chỉ số phụ Precision thể hiện độ chính xác trên các dữ liệu có dự đoán là 1, còn chỉ số phụ Recall thể hiện độ chính xác trên các dữ liệu có nhãn đúng là 1. Chỉ số F1 được sử dụng để có đánh giá tốt hơn chỉ số Accuracy trong các trường hợp mất cân bằng số lượng giữa các nhãn, ví dụ tập dữ liệu có 900 tin giả và 100 tin thật thì chỉ cần dự đoán tất cả là tin giả cũng đã đạt được Accuracy 90% mà không cần đến mô hình nào.

**Bảng 3.2:** Ma trận nhầm lẫn.

		Dự đoán	
		1	0
Nhãn đúng	1	TP	FN
	0	FP	TN

### 3.3.3 Các mô hình được sử dụng

Các mô hình được sử dụng trong các thí nghiệm là: GNNCL [9], BiGCN [32]; các framework UPFD lần lượt dùng GCN [23], GCNFN [8], GAT [24] và GraphSAGE [30] làm mô hình GNN, tạo ra các framework UPFD-GCN, UPFD-GCNFN, UPFD-GAT, và UPFD-SAGE. Ngoại trừ GCNFN, các mô hình khác đều đã được trình bày ở Chương 2. GCNFN là mô hình phát hiện tin giả đầu tiên được xây dựng chỉ trên các lớp GCN, dùng embedding văn bản từ bình luận và hồ sơ người dùng làm đặc trưng. Trên thực tế còn có nhiều mô hình khác tận dụng các thông tin thêm như hình ảnh mà không có trong tập dữ liệu FakeNewsNet, do đó để đảm bảo so sánh công bằng chỉ có các mô hình với mã hóa nội dung bản tin, bình luận người dùng, và đồ thị lan truyền được sử dụng.

### 3.3.4 Chi tiết triển khai thí nghiệm

Các thí nghiệm được triển khai bằng ngôn ngữ lập trình Python và thư viện PyTorch, trong đó tất cả các mô hình GNN đều được triển khai bằng thư

viện PyTorch-Geometric [41]. Kích thước embedding và kích thước batch đều được đặt là 128. Thuật toán tối ưu Adam [42] được sử dụng để cập nhật tham số mô hình, trong đó tốc độ học và trọng số regularization đều được đặt là 0.001. Tỉ lệ chia tập huấn luyện / xác thực / kiểm tra là 70%/20%/10%. Các mô hình UPFD-GCN, UPFD-GCNFN, UPFD-GAT, và UPFD-SAGE lần lượt có 1 lớp GCN, 2 lớp GCN, 1 lớp GAT, và 1 lớp GraphSAGE; và các loại UPFD đều dùng các véc-tơ BERT 768 chiều làm đặc trưng, ngoại trừ UPFD-GCFNF dùng đặc trưng spaCy 300 chiều. BiGCN dùng profile người dùng 10 chiều làm đặc trưng; tỉ lệ DropEdge và Dropout trong BiGCN lần lượt là 0.2 và 0.5. GNNCL sử dụng DiffPool [43] làm bộ mã hóa đồ thị và profile người dùng 10 chiều làm đặc trưng. Các kết quả thí nghiệm được lấy trung bình trên năm lần chạy. Số lượng epochs tối đa là 1000 và cơ chế Kết thúc sớm (Early stopping) được sử dụng, trong đó sẽ dừng huấn luyện nếu chỉ số Accuracy trên tập xác thực không có sự tăng lên nào sau 20 epochs. Các thí nghiệm được thực hiện trên của máy tính cá nhân với CPU Intel Core-i7 1260P 2.1GHz và bộ nhớ RAM 16GB.

### 3.4 Kết quả thí nghiệm

#### 3.4.1 So sánh hiệu năng

Bảng 3.3 thể hiện hiệu năng phát hiện tin giả của sáu mô hình. Trên cả hai tập dữ liệu, mô hình UPFD-SAGE đều vượt trội hơn các mô hình còn lại, khẳng định sự hiệu quả của framework UPFD và thuật toán SAGE. Tất cả mô hình UPFD đều có hiệu năng cao hơn các mô hình khác mà chỉ mã hóa nội dung bản tin mà không xét đến các bài đăng cũ trong lịch sử người dùng, điều đó chứng minh rằng dùng các bài đăng cũ để thể hiện sở thích nội sinh của người dùng đã cải thiện hiệu năng phát hiện tin giả. Đối với SAGE, nó được thiết kế là một framework quy nạp có thể sinh embedding từ các nút lân cận của một nút, nó có khả năng tổng quát hóa tốt hơn trên các nút chưa được thấy. Ngoài ra, ba mô hình UPFD-GCN, UPFD-GCNFN và UPFD-GAT không có nhiều sự khác biệt về hiệu năng, cho thấy hai lớp GCN hoặc GAT có độ hiệu quả tương tự nhau trong các mô hình GNN.

#### 3.4.2 Nghiên cứu sự cắt bỏ

Phần này trình bày nghiên cứu sự cắt bỏ (ablation study), trong đó ta gỡ bỏ hoặc thay thế một thành phần của mô hình và so sánh hiệu năng các biến thể được tạo ra, từ đó thấy được vai trò hay độ quan trọng của thành phần đó

**Bảng 3.3:** So sánh hiệu năng phát hiện tin giả giữa các mô hình.

Mô hình	Politifact		Gossipcop	
	ACC	F1	ACC	F1
GNNCL	74.2	81.82	94.14	94.42
BiGCN	86.83	86.97	92.49	92.69
UPFD-GCN	87.2	87.24	96.52	96.68
UPFD-GCNFN	87.16	87.32	96.34	96.46
UPFD-GAT	87.1	87.5	97.61	97.76
UPFD-SAGE	<b>88.26</b>	<b>88.79</b>	<b>97.85</b>	<b>97.92</b>

trong mô hình.

### a, Các bộ mã hóa embedding

Ta có thể dùng các bộ mã hóa văn bản khác nhau để mã hóa thông tin ngoại sinh và nội sinh. Bảng 3.4 so sánh hiệu năng các mô hình UPFD dùng ba đặc trưng nút khác nhau. Chú ý rằng spaCy và BERT thể hiện các đặc trưng mã hóa sở thích nội sinh của người dùng còn Profile (hồ sơ người dùng) được coi là cơ sở tham chiếu. Kết quả cho thấy các đặc trưng nội sinh spaCy và BERT luôn tốt hơn đặc trưng Profile chỉ mã hóa thông tin hồ sơ người dùng. BERT là đặc trưng tốt nhất để mã hóa văn bản, điều này đã được chứng minh trên các tác vụ NLP khác [27]. Ngoài ra, hiệu năng của BERT còn có thể được nâng cao hơn nữa thông qua tinh chỉnh, đây có thể được dùng làm nghiên cứu trong tương lai.

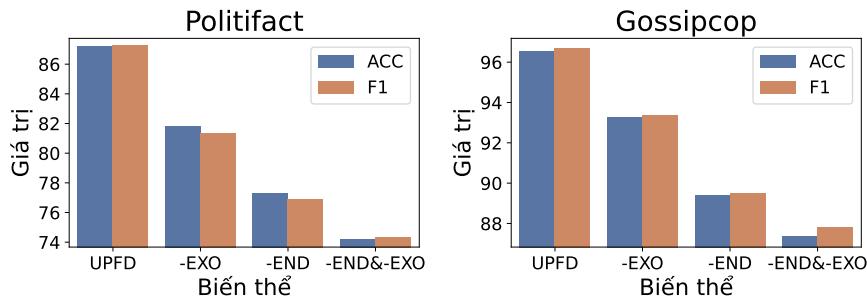
### b, Các biến thể UPFD

Để khẳng định vai trò của sở thích nội sinh và ngữ cảnh ngoại sinh, ta cố định bộ mã hóa văn bản và đồ thị, và thiết kế ba biến thể UPFD trong đó gỡ bỏ thông tin nội sinh, ngoại sinh hoặc cả hai. Cụ thể, BERT được dùng làm bộ mã hóa văn bản và GraphSAGE được dùng làm mô hình GNN trong UPFD. Biến thể UPFD không có thông tin ngoại sinh (-EXO) được triển khai bằng cách gỡ bỏ tất cả các cạnh trong đồ thị lan truyền. Do đó, -EXO mã hóa embedding tin tức chỉ dựa trên đặc trưng nút mà không có trao đổi thông tin giữa các nút. Biến thể UPFD không có thông tin nội sinh (-END) dùng hồ sơ người dùng làm đặc trưng nút và không có thông tin sở thích nội sinh. Biến thể UPFD không có cả thông tin nội sinh và ngoại sinh (-END&-EXO) thay đặc trưng của -EXO bằng đặc trưng hồ sơ người dùng.

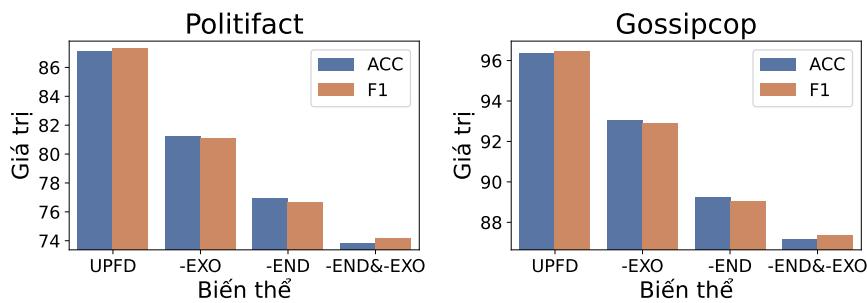
**Bảng 3.4:** Hiệu năng phát hiện tin giả trên các bộ mã hóa văn bản khác nhau.

Mô hình	Bộ mã hóa	Politifact		Gossipcop	
		ACC	F1	ACC	F1
UPFD-GCN	Profile	70.97	70.97	93.41	93.62
	spaCy	83.87	85.71	96.88	97.05
	BERT	<b>87.2</b>	<b>87.24</b>	<b>96.52</b>	<b>96.68</b>
UPFD-GCNFN	Profile	80.65	82.35	90.11	89.71
	spaCy	87.09	87.89	96.23	96.34
	BERT	<b>87.16</b>	<b>88.12</b>	<b>96.34</b>	<b>96.46</b>
UPFD-GAT	Profile	61.29	57.14	93.59	93.75
	spaCy	74.14	75.0	96.34	96.46
	BERT	<b>87.1</b>	<b>87.5</b>	<b>97.61</b>	<b>97.76</b>
UPFD-SAGE	Profile	77.42	79.99	94.14	94.18
	spaCy	79.84	82.68	97.25	97.24
	BERT	<b>88.26</b>	<b>88.79</b>	<b>97.85</b>	<b>97.92</b>

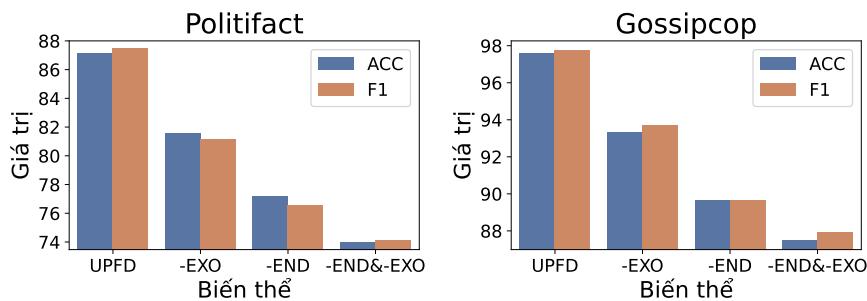
Hình 3.2 so sánh hiệu năng phát hiện tin giả của các biến thể UPFD. Ta có thể thấy việc gỡ bỏ một trong hai thành phần của UPFD sẽ giảm hiệu năng của framework. Ngoài ra, rõ ràng là thông tin nội sinh đóng góp nhiều hơn vào hiệu năng mô hình hơn thông tin ngoại sinh, vì biến thể -END bị tụt hiệu năng hơn nhiều so với biến thể -EXO. Nhìn chung, kết quả thí nghiệm khẳng định việc mã hóa cả thông tin nội sinh và ngoại sinh sẽ cho kết quả tốt nhất.



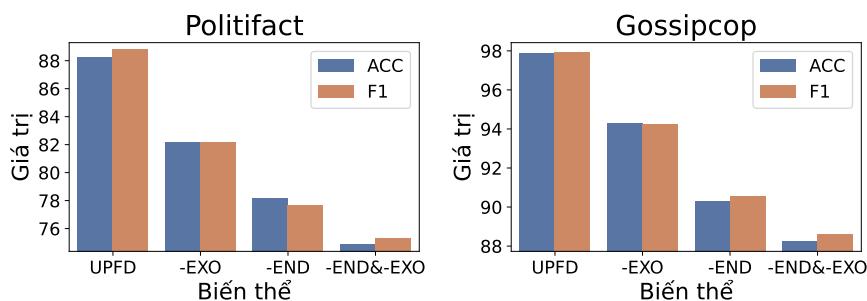
(a) UPFD-GCN



(b) UPFD-GCNFN



(c) UPFD-GAT



(d) UPFD-SAGE

**Hình 3.2:** Hiệu năng phát hiện tin giả trên các biến thể UPFD khác nhau.

## KẾT LUẬN

### Kết luận

Trong luận văn Thạc sĩ này, em đã trình bày một số mô hình ngôn ngữ và triển khai hệ thống xác thực thông tin dựa trên sở thích người dùng UPFD, trong đó sử dụng cả thông tin sở thích nội sinh của người dùng và ngữ cảnh ngoại sinh. Luận văn này đã tổng hợp và so sánh các mô hình từ sáu bài báo khoa học gần đây trong lĩnh vực xác thực thông tin để có cái nhìn tổng quan về các công nghệ mới nhất. Bốn mô hình GNN khác nhau (GCN, GCNFN, GAT, GraphSAGE) đã được sử dụng trong framework UPFD, ngoài ra hai mô hình phát hiện tin giả độc lập khác (GNNCL, BiGCN) cũng được triển khai để so sánh độ hiệu quả. Em đã tiến hành thí nghiệm trên hai tập dữ liệu xác minh tin giả, trong đó lịch sử bài đăng chia sẻ của người dùng trên mạng xã hội Twitter được thu thập làm thông tin nội sinh. Kết quả thí nghiệm cho thấy GraphSAGE là mô hình GNN hiệu quả nhất, và BERT là bộ mã hóa văn bản tốt nhất để sinh đặc trưng nút. Phần thí nghiệm sự cắt bỏ cũng cho thấy vai trò quan trọng của sở thích nội sinh của người dùng trong phát hiện tin giả, và việc gỡ bỏ thông tin nội sinh hoặc ngoại sinh đều làm giảm hiệu năng của hệ thống.

### Hướng phát triển

Trong tương lai, nghiên cứu về phát hiện tin giả có thể tiếp tục phát triển bằng triển khai các mô hình GNN hoặc framework khác. Ngoài ra bộ mã hóa văn bản BERT có thể được tinh chỉnh tiếp dựa trên nội dung bản tin để cung cấp embedding tốt hơn cho đặc trưng nút.

## CHỈ MỤC

- BERT, 9, 16  
bidirectional cross attention, 18  
BiGCN, 25  
bộ giải mã, 9  
bộ mã hóa, 9  
cơ chế attention, 9  
cạnh, 4  
dot-product attention, 11  
EWC, 29  
GAE, 7  
GAT, 8, 21  
GCN, 8, 20  
GEM, 29  
GNN, 2, 6  
GNNCL, 28  
GraphSAGE, 23  
hàm kích hoạt, 7  
hàm lan truyền, 7  
hàm tổng hợp, 23  
inductive, 23  
logit, 32  
LSTM, 25  
ma trận kề, 4, 6  
ma trận kề khôi phục, 8  
ma trận trọng số, 7  
ma trận đặc trưng, 6  
masked LM, 18  
max-pooling, 25  
mean-pooling, 28  
multi-head attention, 12  
mạng nơ-ron truyền thẳng, 11  
ngữ cảnh ngoại sinh, 30  
NLI, 18  
NLP, 9  
nút, 4  
positional encoding, 16  
QA, 18  
RNN, 8  
scaled dot-product attention, 12  
self-attention, 11  
STGNN, 8  
sở thích nội sinh, 30  
tin giả, 1  
tinh chỉnh, 16  
tiền huấn luyện, 16  
token, 16, 17  
transductive, 23  
Transformer, 9, 12  
tác vụ cấp cạnh, 6  
tác vụ cấp nút, 6  
tác vụ cấp đồ thị, 6  
UPFD, 30  
WordPiece embeddings, 17  
điểm căn chỉnh, 11  
đô thị, 4  
đô thị có hướng, 4  
đô thị vô hướng, 4

## TÀI LIỆU THAM KHẢO

- [1] H. T. Phan, N. T. Nguyen, and D. Hwang, “Fake news detection: A survey of graph neural network methods,” *Applied Soft Computing*, p. 110 235, 2023.
- [2] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich, “A review of relational machine learning for knowledge graphs,” *Proceedings of the IEEE*, vol. 104, no. 1, pp. 11–33, 2015.
- [3] B. Shi and T. Weninger, “Fact checking in heterogeneous information networks,” in *Proceedings of the 25th International Conference Companion on World Wide Web*, 2016, pp. 101–102.
- [4] X. Zhou and R. Zafarani, “Fake news: A survey of research, detection methods, and opportunities,” *arXiv preprint arXiv:1812.00315*, vol. 2, 2018.
- [5] L. Wu and H. Liu, “Tracing fake-news footprints: Characterizing social media messages by how they propagate,” in *Proceedings of the eleventh ACM international conference on Web Search and Data Mining*, 2018, pp. 637–645.
- [6] J. Ma, W. Gao, and K.-F. Wong, “Detect rumors in microblog posts using propagation structure via kernel learning,” Association for Computational Linguistics, 2017.
- [7] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, “A comprehensive survey on graph neural networks,” *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.
- [8] F. Monti, F. Frasca, D. Eynard, D. Mannion, and M. M. Bronstein, “Fake news detection on social media using geometric deep learning,” *arXiv preprint arXiv:1902.06673*, 2019.
- [9] Y. Han, S. Karunasekera, and C. Leckie, “Graph neural networks with continual learning for fake news detection from social media,” *arXiv preprint arXiv:2007.03316*, 2020.
- [10] V.-H. Nguyen, K. Sugiyama, P. Nakov, and M.-Y. Kan, “Fang: Leveraging social context for fake news detection using graph representation,” in *Proceedings of the 29th ACM international conference on information & knowledge management*, 2020, pp. 1165–1174.

- [11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.
- [12] W. Shi and R. Rajkumar, “Point-gnn: Graph neural network for 3d object detection in a point cloud,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 1711–1719.
- [13] G. Chen, Y. Tian, and Y. Song, “Joint aspect extraction and sentiment analysis with directional graph convolutional networks,” in *Proceedings of the 28th international conference on computational linguistics*, 2020, pp. 272–279.
- [14] C. Zhang, Q. Li, and D. Song, “Aspect-based sentiment classification with aspect-specific graph convolutional networks,” *arXiv preprint arXiv:1909.03477*, 2019.
- [15] J. Bastings, I. Titov, W. Aziz, D. Marcheggiani, and K. Sima’an, “Graph convolutional encoders for syntax-aware neural machine translation,” *arXiv preprint arXiv:1704.04675*, 2017.
- [16] D. Marcheggiani, J. Bastings, and I. Titov, “Exploiting semantics in neural machine translation with graph convolutional networks,” *arXiv preprint arXiv:1804.08313*, 2018.
- [17] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, *et al.*, “Convolutional networks on graphs for learning molecular fingerprints,” *Advances in neural information processing systems*, vol. 28, 2015.
- [18] B. Sanchez-Lengeling, E. Reif, A. Pearce, and A. B. Wiltschko, “A gentle introduction to graph neural networks,” *Distill*, 2021, <https://distill.pub/2021/gnn-intro>. DOI: 10.23915/distill.00033.
- [19] M. Allamanis, M. Brockschmidt, and M. Khademi, “Learning to represent programs with graphs,” *arXiv preprint arXiv:1711.00740*, 2017.
- [20] G. Lample and F. Charton, “Deep learning for symbolic mathematics,” *arXiv preprint arXiv:1912.01412*, 2019.
- [21] T. N. Kipf and M. Welling, “Variational graph auto-encoders,” *arXiv preprint arXiv:1611.07308*, 2016.
- [22] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, “Structured sequence modeling with graph convolutional recurrent networks,” in

*Neural Information Processing: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13-16, 2018, Proceedings, Part I* 25, Springer, 2018, pp. 362–373.

- [23] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” *arXiv preprint arXiv:1609.02907*, 2016.
- [24] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, *et al.*, “Graph attention networks,” *stat*, vol. 1050, no. 20, pp. 10–48 550, 2017.
- [25] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *arXiv preprint arXiv:1409.0473*, 2014.
- [26] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.
- [27] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.
- [28] Y. Wu, M. Schuster, Z. Chen, *et al.*, “Google’s neural machine translation system: Bridging the gap between human and machine translation,” *arXiv preprint arXiv:1609.08144*, 2016.
- [29] Y. Zhu, R. Kiros, R. Zemel, *et al.*, “Aligning books and movies: Towards story-like visual explanations by watching movies and reading books,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 19–27.
- [30] W. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” *Advances in neural information processing systems*, vol. 30, 2017.
- [31] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [32] T. Bian, X. Xiao, T. Xu, *et al.*, “Rumor detection on social media with bi-directional graph convolutional networks,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, 2020, pp. 549–556.
- [33] Y. Rong, W. Huang, T. Xu, and J. Huang, “Dropedge: Towards deep graph convolutional networks on node classification,” in *International Conference on Learning Representations*, 2019.

- [34] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.
- [35] D. Lopez-Paz and M. Ranzato, “Gradient episodic memory for continual learning,” *Advances in neural information processing systems*, vol. 30, 2017.
- [36] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, *et al.*, “Overcoming catastrophic forgetting in neural networks,” *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.
- [37] Y. Dou, K. Shu, C. Xia, P. S. Yu, and L. Sun, “User preference-aware fake news detection,” in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 2051–2055.
- [38] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv:1301.3781*, 2013.
- [39] M. Honnibal and I. Montani, “Spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing,” *To appear*, vol. 7, no. 1, pp. 411–420, 2017.
- [40] K. Shu, D. Mahudeswaran, S. Wang, D. Lee, and H. Liu, “Fakenewsnet: A data repository with news content, social context and dynamic information for studying fake news on social media,” *arXiv preprint arXiv:1809.01286*, 2018.
- [41] M. Fey and J. E. Lenssen, “Fast graph representation learning with pytorch geometric,” *arXiv preprint arXiv:1903.02428*, 2019.
- [42] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [43] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, “Hierarchical graph representation learning with differentiable pooling,” *Advances in neural information processing systems*, vol. 31, 2018.