

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



Hoàng Minh Thanh - Phan Minh Tâm

**BÁO CÁO KHÓA LUẬN TỐT NGHIỆP DỰ ĐOÁN
LIÊN KẾT TRONG ĐỒ THỊ TRỊ THỨC**

**KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN
CHƯƠNG TRÌNH HOÀN CHỈNH**

Tp. Hồ Chí Minh, tháng 10/2020

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

Hoàng Minh Thanh - 18424062

Phan Minh Tâm - 18424059

**BÁO CÁO KHÓA LUẬN TỐT NGHIỆP DỰ ĐOÁN
LIÊN KẾT TRONG ĐỒ THỊ TRI THỨC**

**KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN
CHƯƠNG TRÌNH HOÀN CHỈNH**

GIÁO VIÊN HƯỚNG DẪN

ThS. Lê Ngọc Thành

BM. Khoa Học Máy Tính

Tp. Hồ Chí Minh, tháng 10/2020

LỜI CAM ĐOAN

Tôi xin cam đoan đây là công trình nghiên cứu của riêng chúng tôi. Các số liệu và kết quả nghiên cứu trong luận văn này là trung thực và không trùng lặp với các đề tài khác.

LỜI CẢM ƠN

Chúng tôi xin chân thành cảm ơn thầy Lê Ngọc Thành đã tận tình hướng dẫn, truyền đạt kiến thức và kinh nghiệm, và đưa ra các giải pháp cho chúng tôi trong suốt quá trình thực hiện đề tài luận văn tốt nghiệp này.

Xin gửi lời cảm ơn đến quý thầy cô Khoa Công Nghệ Thông Tin trường Đại Học Khoa Học Tự Nhiên - Đại Học Quốc Gia Thành Phố Hồ Chí Minh, những người đã truyền đạt kiến thức quý báu cho chúng tôi trong thời gian học tập vừa qua.

Đồng thời cảm ơn các nhà khoa học đã nghiên cứu về đề tài mà chúng tôi đã trích dẫn để có thể có những kiến thức hoàn thiện luận văn của chúng tôi.

Sau cùng chúng tôi xin gửi lời cảm ơn đến gia đình, bạn bè,... những người luôn động viên, giúp đỡ chúng tôi trong quá trình làm luận văn.

Một lần nữa, xin chân thành cảm ơn !

Mục lục

Lời cam đoan	1
Lời cảm ơn	2
Danh mục hình ảnh	5
Danh mục bảng	6
Danh mục thuật toán	7
Danh mục các ký hiệu, các chữ viết tắt	8
Bảng chú thích thuật ngữ	10
Tóm tắt	1
Chương 1 Giới thiệu	1
Chương 2 Các công trình liên quan	3
2.1 Định nghĩa đồ thị tri thức	3
2.2 Dự đoán liên kết trong đồ thị tri thức	4
2.3 Các lĩnh vực nghiên cứu về đồ thị tri thức	6
Chương 3 Phương pháp dựa trên luật	9
3.1 Luật Horn	9
3.2 Định nghĩa ngôn ngữ đồ thị logic	10
3.3 Thuật toán AnyBURL	13
3.3.1 AnyBURL	13
3.3.2 Tạo luật	14
3.4 Thuật toán AnyBURL mở rộng	15
3.4.1 Thuật toán 3 học offline-to-online	15
3.4.2 Thuật toán 4 học online-to-online	17
Chương 4 Phương pháp dựa trên học sâu	18
4.1 Nhúng đồ thị	18
4.1.1 Phương pháp thiết lập bài toán nhúng đồ thị	21

4.1.2	Các kỹ thuật nhúng đồ thị	24
4.2	Cơ chế chú ý đa đỉnh	32
4.2.1	Cơ Chế Chú Ý (Attention Mechanism)	32
4.2.2	Chú Ý Đa Đỉnh (Multi-Head Attention)	33
4.3	Mạng đồ thị chú ý	34
4.4	Mô hình KBGAT	37
4.4.1	Khởi tạo vector nhúng	38
4.4.2	Mô hình mã hóa	41
4.4.3	Mô hình dự đoán ConvKB	44
Chương 5	Thực nghiệm	46
5.1	Các tập dữ liệu huấn luyện	46
5.1.1	Bộ dữ liệu FB15k	47
5.1.2	Bộ dữ liệu FB15k-237	48
5.1.3	Bộ dữ liệu WN18	48
5.1.4	Bộ dữ liệu WN18RR	49
5.2	Các độ đo	49
5.3	Phương pháp huấn luyện	50
5.3.1	Huấn luyện trên mô hình KBGAT	50
5.4	Kết quả thực nghiệm	51
Chương 6	Kết luận	54
TÀI LIỆU TRÍCH DẪN		56
Chương A	Các siêu tham số tối ưu	62

DANH MỤC HÌNH ẢNH

Hình 2.1:	Example of an input graph	3
Hình 2.2:	Danh mục các lĩnh vực nghiên cứu trên đồ thị tri thức . . .	7
Hình 3.1:	Ví dụ đồ thị tri thức	12
Hình 4.1:	Các kỹ thuật nhúng đồ thị	20
Hình 4.2:	Node embedding with each vector representing node features	21
Hình 4.3:	Edge embedding with each vector representing edge features	22
Hình 4.4:	Embedding a graph substructure	23
Hình 4.5:	Whole-graph embedding	23
Hình 4.6:	Các kỹ thuật nhúng đồ thị	24
Hình 4.7:	Đồ thị tri thức và các hệ số chú ý chuẩn hóa của thực thể .	34
Hình 4.8:	Minh họa về các vector nhúng trong mô hình TransE	38
Hình 4.9:	TransE embedding model	40
Hình 4.10:	Minh họa các lớp mã hóa của mô hình KBGAT	41
Hình 4.11:	Minh họa các lớp giải mã của mô hình ConvKB với 3 filter .	44

DANH MỤC BẢNG

Bảng 4.1: Bảng so sánh ưu và nhược điểm của kỹ thuật nhúng đồ thị .	30
Bảng 5.1: Thông tin các tập dữ liệu	47
Bảng 5.2: Kết quả thực nghiệm trên tập FB15k, FB15k-237	52
Bảng 5.3: Kết quả thực nghiệm trên tập WN18, WN18RR	52
Bảng 5.4: Kết quả độ chính xác hai chiến lược thêm tri thức mới . . .	53
Bảng 5.5: Kết quả đánh giá về số luật hai chiến lược thêm tri thức mới	53

DANH MỤC THUẬT TOÁN

Thuật toán 1: Anytime Bottom-up Rule Learning	13
Thuật toán 2: Generate Rules(p)	14
Thuật toán 3: BatchAnyBURL Learning batch size	16
Thuật toán 4: EdgeAnyBURL	17
Thuật toán 5: Thuật toán học vector nhúng TransE [5]	39

DANH MỤC CÁC KÝ HIỆU, CÁC CHỮ VIẾT TẮT

Danh sách Ký hiệu

Ký hiệu	Mô tả
\mathcal{G}	Đồ thị
$\mathcal{G}_{\text{mono}}$	Đồ thị đồng nhất
$\mathcal{G}_{\text{hete}}$	Đồ thị không đồng nhất
$\mathcal{G}_{\text{know}}$	Đồ thị tri thức
V, E	Tập hợp đỉnh, Tập hợp cạnh
e, e_i	Thực thể, thực thể thứ i
r, r_k	Quan hệ, quan hệ thứ k
t_{ijk}, t_{ij}^k	Một cạnh/bộ ba
\vec{e}, \vec{r}	Thực thể nhúng, quan hệ nhúng
$\langle h, r, t \rangle$	Bộ ba gồm thực thể đỉnh (head), quan hệ (relation), thực thể đuôi (tail)
T^v, T^e	Tập hợp loại đỉnh, loại cạnh
N_e, N_r	Số lượng tập thực thể, tập quan hệ
N_{head}	Số lượng đỉnh tự chú ý
\mathbb{R}	Tập số thực
\mathbf{E}, \mathbf{R}	Ma trận nhúng thực thể, quan hệ
\mathbf{S}	Tập dữ liệu huấn luyện
$*$	Phép tính tích chập
σ	Hàm biến đổi phi tuyến tính
\mathbf{W}	Ma trận trọng số
$\ _{k=1}^K$	Phép ghép chồng từ 1 đến K lớp
$\ $	Phép ghép chồng
\cdot^T	Phép chuyển vị
$\ W\ _2^2$	Chuẩn hóa L2
\vee, \wedge	Phép hội, phép giao
\oplus	Phép toán hai ngôi
\cap	Phép hợp
\neg	Phép phủ định

$\bigwedge_{i=1}^n$	Phép nối liền
γ	Biên lề
μ	Tốc độ học
ω	Số lượng lớp tích chập
Ω	Bộ lọc tính tích chập

BẢNG CHÚ THÍCH THUẬT NGỮ

Thuật ngữ	Ý nghĩa đầy đủ
AnyBURL	Anytime Bottom-up Rule Learning - Thuật toán học luật từ dưới lên bất kỳ lúc nào
Knowledge Graph ($\mathcal{G}_{\text{know}}$)	Đồ thị tri thức - Tập hợp các ground atoms hoặc facts
ConvE / ConvKB	Mô hình CNN cho nhúng đồ thị - Biến thể dùng vector nhúng cho $\langle h, r, t \rangle$
Graph Attention Network (GAT)	Mạng đồ thị chú ý - Sử dụng attention để tổng hợp thông tin từ lân cận
KBGAT	GAT có vector nhúng quan hệ - Biến thể GAT kết hợp thêm nhúng quan hệ
Saturation (SAT)	Độ bão hòa - Tỷ lệ luật mới học được so với luật đã học

Tóm tắt

Đồ thị tri thức là cấu trúc giúp biểu diễn thông tin trong thế giới thực đã được Google nghiên cứu và phát triển cho công cụ tìm kiếm của mình rất thành công [15]. Việc khai thác đồ thị tri thức không chỉ có truy vấn, phân tích mà còn hoàn thiện những thông tin còn thiếu cũng như dự đoán liên kết dựa trên những thông tin sẵn có trên đồ thị tri thức. Chính vì vậy, trong báo cáo này chúng tôi sẽ trình bày cơ bản về đồ thị tri thức và hai phương pháp để dự đoán liên kết trong đồ thị là phương pháp dựa trên luật và phương pháp dựa trên học sâu. Với phương pháp dựa trên luật, chúng tôi dựa trên mô hình AnyBURL và cung cấp thêm 2 chiến lược thêm tri thức mới vào đồ thị.

Với phương pháp học sâu, chúng tôi trình bày lại cơ chế chú ý trong xử lý ngôn ngữ tự nhiên, từ đó được áp dụng vào đồ thị tri thức và trình bày lại đầy đủ mô hình cải tiến là mô hình KBGAT. Bằng cách ghép chồng các lớp với nhau mà những đỉnh có thể chú ý với những đặc trưng lân cận mà không tốn chi phí tính toán nào hoặc phụ thuộc vào việc biết trước cấu trúc đồ thị trước đó. Hai mô hình của chúng tôi đạt được kết quả tốt hơn đáng kể so với các phương pháp dự đoán liên kết khác được áp dụng trên bốn tập dữ liệu chuẩn.

Chương 1. GIỚI THIỆU

Ngày nay đồ thị đã được ứng dụng vào mọi mặt của đời sống, với đồ thị về mạng xã hội (Facebook [44]) thể hiện thông tin kết nối từng người với nhau, những nơi chúng ta đến, những thông tin chúng ta tương tác, đồ thị cũng được sử dụng làm cấu trúc trong hệ thống gợi ý video (Youtube [3]), trong mạng các chuyến bay, hệ thống định vị GPS, những tính toán khoa học hay thậm chí là kết nối não. Đồ thị tri thức của Google (Google's Knowledge Graph [15]) được Google giới thiệu năm 2012 [22], một loại đồ thị biểu diễn thông tin, là một trong những ứng dụng rõ ràng nhất về đồ thị tri thức cũng như cách dữ liệu được khai thác và biểu diễn trên đồ thị tri thức.

Khai thác đồ thị tri thức hiệu quả cung cấp cho người dùng hiểu sâu hơn về những gì đằng sau dữ liệu và từ đó có thể mang lại lợi ích cho nhiều ứng dụng trong thực tế. Tuy nhiên trong thực tế, luôn có những tri thức mới được sinh ra mỗi ngày, thông tin thu được thường bị mất mát và không đầy đủ, từ đó nảy sinh ra vấn đề hoàn thiện đồ thị tri thức (knowledge graph completion) hay dự đoán liên kết (linking prediction) trong đồ thị tri thức. Hầu hết các phương pháp tiếp cận hiện nay là dự đoán một cạnh mới nối từ đỉnh này tới đỉnh khác. Với cách tiếp cận như vậy đồ thị tri thức có đầy đủ các tri thức nói cách khác làm cho đồ thị dày đặc nhờ tạo thêm các cạnh nối. Nhưng như vậy chỉ mới giải quyết được vấn đề hoàn thành đồ thị, vấn đề thêm một (hoặc một lượng) tri thức mới vào đồ thị vẫn còn là một câu hỏi mở. Hiện nay các bài toán liên quan đến hoàn thành đồ thị tri thức có hai cách tiếp cận chính là tối ưu hóa hàm mục tiêu tức là đưa ra dự đoán dựa trên có sai sót ít nhất như trong RuDiK[34], AMIE[14], RuleN[29] liên quan tới các ứng dụng phân loại đỉnh, phân loại cạnh. Hoặc đưa ra một danh sách gồm k ứng viên với số điểm đại diện cho độ tin cậy giảm dần như trong các nghiên cứu TransE[5], ConvKB[48] liên quan tới các hệ thống gợi ý (recommend system)...Cách tiếp cận của chúng tôi dựa trên cách tạo ra một danh sách gồm k ứng viên này.

Với mỗi cách tiếp cận trên có hai phương pháp chính đưa ra nghiên cứu một cách dựa trên luật như trong AnyBURL[27] hoặc dựa trên nhúng đồ thị như trong ConvE[11], TransE[5], ComplEx [43]. Với mong muốn được tiếp cận các phương pháp có hệ thống nên chúng tôi chọn ở cả hai phương pháp để nghiên cứu thực hiện đề tài này. Đối với phương pháp dựa trên luật chúng tôi chọn phương pháp AnyBURL[27] còn với phương pháp dựa trên nhúng đồ thị chúng tôi chọn phương pháp KBAT [32] sử dụng cơ chế chú ý.

Đóng góp của chúng tôi trong phương pháp AnyBURL[27] gồm mã nguồn Python ¹, cùng với đó chúng tôi cung cấp thêm hai chiến lược để thêm tri thức mới vào đồ thị mà chúng tôi gọi là online-to-offline là mở rộng của AnyBURL trong việc tạo ra các luật khi có một lượng (tập hợp) tri thức mới được thêm vào. Online-to-online sẽ tạo ngay các luật mới khi có một tri thức mới(cạnh) được thêm vào. Với phương pháp dựa trên nhúng đồ thị chúng tôi sẽ trình bày lại về cơ chế chú ý (attention mechanisms [45]), cách cơ chế chú ý được áp dụng vào đồ thị tri thức bằng mô hình Mạng Đồ Thị Chú Ý (Graph Attention Network [46]), mô hình KBGAT [32]. Đóng góp của chúng tôi trong phương pháp học sâu bao gồm mã nguồn trên Github và quá trình huấn luyện trên Colab được công khai ở mã nguồn github ² về quá trình huấn luyện và kết quả của mô hình.

¹<https://github.com/MinhTamPhan/mythesis>

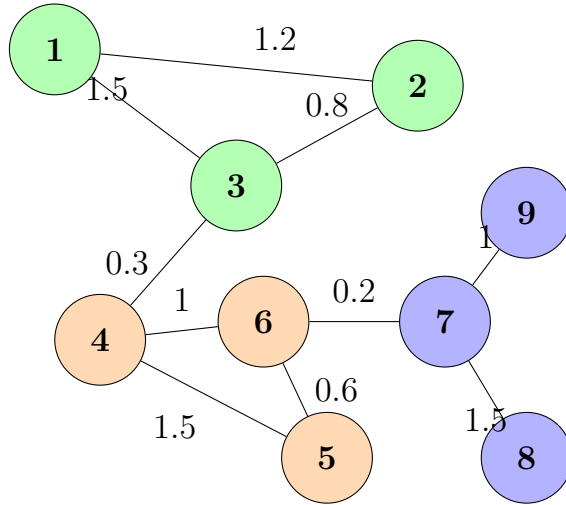
²<https://github.com/hmthanh/GCAT>

Chương 2. CÁC CÔNG TRÌNH LIÊN QUAN

Trong phần này chúng tôi sẽ trình bày về các định nghĩa cơ bản về đồ thị tri thức, để từ đó hiểu được nhiệm vụ dự đoán liên kết trong đồ thị tri thức là gì cũng như các hướng nghiên cứu bên khác liên quan về đồ thị tri thức.

2.1 Định nghĩa đồ thị tri thức

Các định nghĩa cơ bản về đồ thị tri thức được nhóm tác giả Cai, Hongyun[7] và Goyal, Palash[16] tổng hợp và phân loại như sau :



Hình 2.1: Example of an input graph

- **Định nghĩa 1 (Đồ thị)** $\mathcal{G} = (V, E)$, trong đó $v \in V$ là một đỉnh và $e \in E$ là một cạnh. \mathcal{G} được liên kết với hàm ánh xạ loại đỉnh $f_v : V \rightarrow T^v$ và hàm ánh xạ loại cạnh: $f_e : E \rightarrow T^e$.

Trong đó: T^v và T^e lần lượt là tập hợp các loại đỉnh và loại cạnh. Mỗi đỉnh $v_i \in V$ thuộc về một loại cụ thể, tức là, $f_v(v_i) \in T^v$. Tương tự, đối với $e_{ij} \in E$, $f_e(e_{ij}) \in T^e$.

- **Định nghĩa 2 (Đồ thị đồng nhất)** Đồ thị đồng nhất (*homogeneous graph*) : $\mathcal{G}_{homo} = (V, E)$ là đồ thị trong đó $|T^v| = |T^e| = 1$. Tất cả các đỉnh trong \mathcal{G} thuộc về một loại duy nhất và tất cả các cạnh thuộc về một loại duy nhất.
- **Định nghĩa 3 (Đồ thị không đồng nhất)** Đồ thị không đồng nhất (*heterogeneous graph*) : $\mathcal{G}_{hete} = (V, E)$ là một đồ thị trong đó $|T^v| > 1$ hoặc $|T^e| > 1$. Tức là có nhiều hơn một loại đỉnh hoặc nhiều hơn một loại cạnh.
- **Định nghĩa 4 (Đồ thị tri thức)** Đồ thị tri thức (*knowledge graph*) $\mathcal{G}_{know} = (V, R, E)$ là một đồ thị có hướng, có tập đỉnh là biểu diễn cho các thực thể (*entities*), tập quan hệ biểu diễn các mối quan hệ (*relations*) giữa các đỉnh, tập cạnh (*edges*) biểu diễn các sự kiện $E \subseteq V \times R \times V$ là gồm bộ ba *subject-property-object*. Mỗi cạnh là một mẫu gồm (*entity_{head}, relation, entity_{tail}*) (ký hiệu là $\langle h, r, t \rangle$) biểu thị mối quan hệ của r từ thực thể h đến thực thể t .

Trong đó $h, t \in V$ là các thực thể và $r \in R$ là các quan hệ. Chúng ta gọi $\langle h, r, t \rangle$ một bộ ba (triples) đồ thị tri thức.

Ví dụ: trong [Hình 4.7](#) có hai bộ ba: $\langle \text{Tom Cruise, born_in, New York} \rangle$ và $\langle \text{New York, state_of, U.S.} \rangle$. Lưu ý rằng các thực thể và quan hệ trong đồ thị tri thức thường có các loại khác nhau. Do đó, đồ thị tri thức có thể được xem như là một ví dụ của đồ thị không đồng nhất.

2.2 Dự đoán liên kết trong đồ thị tri thức

Dự đoán liên kết (link prediction) hay hoàn thiện đồ thị tri thức (knowledge graph completion) là nhiệm vụ khai thác những sự kiện có sẵn trong đồ thị tri thức để suy luận ra sự kiện còn thiếu. Điều này tương đương với việc đoán đúng thực thể đuôi $\langle h, r, ? \rangle$ (dự đoán đuôi) hoặc $\langle ?, r, t \rangle$ (dự đoán đầu). Để đơn giản, thay vì gọi dự đoán đầu hoặc đuôi, một cách tổng quát chúng ta gọi thực thể nguồn (source) là thực thể đã biết trong việc dự đoán, thực thể đích (target) là cái chúng ta cần dự đoán.

Hầu hết các nghiên cứu hiện tại về việc dự đoán liên kết của đồ thị tri thức đều liên quan đến các phương pháp tiếp cận tập trung vào khái niệm nhúng một đồ thị đã cho trong một không gian vectơ có số chiều thấp. Ngược lại với

các tiếp cận này là một phương pháp dựa trên luật được nghiên cứu trong [27]. Thuật toán cốt lõi của nó dựa trên lấy mẫu một luật bất kỳ, sau đó khái quát thành các quy tắc Horn[25]. Tiếp đó dùng thống kê để tính độ tin cậy của các luật được khái quát. Khi dự đoán một liên kết mới (cạnh mới) của đồ thị chúng ta dự đoán một đỉnh có cạnh nối với một quan hệ cụ thể (label) với đỉnh còn lại hay không. Cũng đã có rất nhiều phương pháp được nghiên cứu, đề xuất để học các luật trong đồ thị chẳng hạn như trong RuDiK[34], AMIE[14], RuleN[29]. Như đã nói trong phần trước có hai cách tiếp cận chính cho bài toán này một là tối ưu hóa hàm mục tiêu. Tìm ra một bộ quy tắc nhỏ bao gồm phần lớn các ví dụ là đúng và ít sai sót nhất có thể như được nghiên cứu trong RuDiK[34]. Còn cách tiếp cận còn lại cũng là cách tiếp cận mà chúng tôi chọn nghiên cứu là cố gắng tìm hiểu mọi quy tắc khả thi có thể sau đó tạo xếp hạng k ứng viên tiềm năng với một độ tin cậy nhất định được đo trên tập huấn luyện.

Phương pháp dựa trên luật của chúng tôi phần lớn dựa vào phương pháp Anytime Bottom-Up Rule Learning for Knowledge Graph Completion [28] mà sau đây chúng tôi gọi là **AnyBURL**. Như tên của phương pháp này phương pháp chủ yếu chú trọng vào vấn đề hoàn thành đồ thị, điền những phần còn thiếu vào đồ thị. Vấn đề tồn đọng lại ở mô hình này khi có một cạnh mới hay một tri thức mới được thêm vào đồ thị sẽ phải đào tạo lại toàn bộ mô hình. Chúng tôi giải quyết vấn đề này theo hai chiến lược offline-to-online tức là khi thêm vào đồ thị tập hợp các cạnh thì mới thực hiện lại quá trình đào tạo lại một phần của đồ thị và chiến lược thứ 2 là online-to-online khi thêm một cạnh mới sẽ thực hiện đào tạo lại ngay một phần có liên quan tới cạnh vừa thêm vào.

Trong nhánh các phương pháp về học sâu, rất nhiều kỹ thuật học sâu thành công trong xử lý ảnh và xử lý ngôn ngữ tự nhiên được áp dụng vào đồ thị tri thức như : Mạng Neural Tích Chập (Convolution Neural Network - CNN [24]), Mạng Neural Hồi Quy (Recurrent Neural Network[20]), và gần đây như Transformer ([50]), Mạng Neural Bao Bọc (Capsule Neural Network - CapsNet [38]). Bên cạnh đó các nghiên cứu còn sử dụng một số kỹ thuật khác như Random Walks, các mô hình dựa trên cấu trúc phân cấp, .. Ưu điểm chung của nhóm các phương pháp học sâu trên đồ thị tri thức đó là tự động rút trích các đặc trưng và có thể khái quát hóa cấu trúc phức tạp của đồ thị dựa trên một lượng lớn dữ liệu huấn luyện. Tuy nhiên, một số phương pháp chỉ chủ yếu tập trung vào cấu trúc

dạng lưới mà không giữ được đặc trưng không gian của đồ thị tri thức. Cơ chế chú ý hay lớp chú ý đa đỉnh (multi-head attention layer) đã được áp dụng vào đồ thị bằng mô hình Mạng Đồ Thị Chú Ý (Graph Attention Network - GAT [46]) giúp tổng hợp thông tin của một thực thể dựa vào trọng số chú ý của thực thể gốc đối với các thực thể lân cận. Tuy nhiên, mô hình đồ thị chú ý lại thiếu thông tin của vector nhúng quan hệ cũng như các vector nhúng lân cận của một thực thể gốc, một phần rất quan trọng giúp thể hiện vai trò của từng thực thể. Vấn đề đó đã được giải quyết trong báo cáo Learning Attention-based Embeddings for Relation Prediction in Knowledge Graphs (**KBAT** [32]), mô hình được chúng tôi chọn làm cơ sở nghiên cứu. Cơ chế chú ý đang là một trong những cấu trúc học sâu đạt được hiệu quả nhất hiện nay (state-of-the-art) vì nó đã được chứng minh là thay thế cho bất kỳ phương pháp tính tích chập nào [9], hơn nữa nó cũng nằm trong cấu trúc cơ bản để áp dụng trên các mô hình mới nhất trên ngôn ngữ tự nhiên như mô hình Megatron-LM [39], và trên phân đoạn hình ảnh như mô hình HRNet-OCR (Hierarchical Multi-Scale Attention [41]). Một số phương pháp thú vị [10] đã cải tiến dựa trên cơ chế chú ý, tuy nhiên nó lại chưa được áp dụng vào đồ thị tri thức, vì vậy chúng tôi chọn nhóm phương pháp này để áp dụng các cải tiến mới nhất vào đồ thị tri thức.

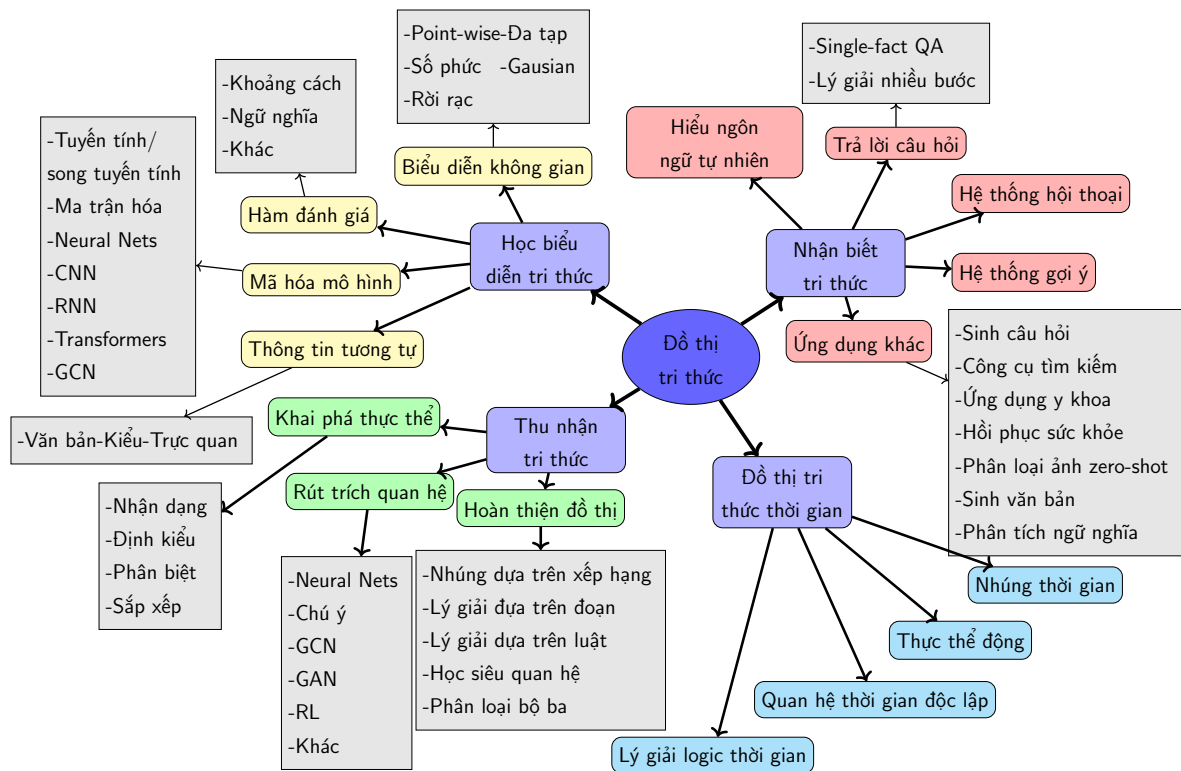
2.3 Các lĩnh vực nghiên cứu về đồ thị tri thức

Biểu diễn tri thức đã từng có lịch sử phát triển suốt chiều dài lịch sử trong lĩnh vực logic và trí tuệ nhân tạo. Trên đồ thị tri thức, có 4 bốn nhóm nghiên cứu chính đã được phân loại và tổng hợp ở báo cáo [22] bao gồm : Học Biểu Diễn Tri Thức (Knowledge Representation Learning), Thu Nhận Tri Thức (Knowledge Acquisition), Đồ Thị Tri Thức Về Thời Gian (Temporal Knowledge Graphs), Ứng Dụng Nhận Biết Tri Thức (Knowledge-aware Applications). Tất cả các danh mục nghiên cứu được minh họa ở [Hình 2.2](#).

Học biểu diễn tri thức

Học biểu diễn tri thức là vấn đề tìm hiểu thiết yếu của đồ thị tri thức giúp mở ra rất nhiều ứng dụng trong thực tế. Học biểu diễn tri thức được phân loại thành bốn nhóm con bao gồm :

- *Biểu Diễn Không Gian* (Representation Space) nghiên cứu về cách các



Hình 2.2: Danh mục các lĩnh vực nghiên cứu trên đồ thị tri thức

thực thể và quan hệ được biểu diễn trong không gian. Biểu diễn không gian bao gồm không gian điểm (point-wise), đa tạp (manifold), không gian vector số phức (complex), phân phối Gaussian và không gian rời rạc.

- *Hàm Đánh Giá* (Scoring Function) nghiên cứu về hàm đo lường giá trị của một bộ ba trong thực tế, bao gồm các hàm đánh giá dựa trên khoảng cách hoặc dựa trên sự tương đồng.
- *Mã Hóa Mô Hình* (Encoding Models) nghiên cứu về cách biểu diễn và học các tương tác giữa các mối quan hệ. Đây là hướng nghiên cứu chính hiện nay, bao gồm các mô hình tuyến tính hoặc phi tuyến tính, phân rã ma trận hoặc mạng neural.
- *Thông Tin Bổ Trợ* (Auxiliary Information) nghiên cứu về cách kết hợp vào các phương pháp nhúng, các thông tin bổ trợ bao gồm văn bản, hình ảnh và loại thông tin.

Thu nhận tri thức

Thu nhận tri thức nghiên cứu về cách thu nhận tri thức dựa trên đồ thị tri

thức, bao gồm hoàn thiện đồ thị (knowledge graph completion), khai thác quan hệ và khai phá thực thể. Khai thác quan hệ và khai phá thực thể là nhóm phương pháp khai thác tri thức mới (bao gồm các quan hệ hoặc thực thể) trong đồ thị từ văn bản. Hoàn thiện đồ thị là nhiệm vụ mở rộng đồ thị tri thức dựa trên đồ thị đang có. Hoàn thiện đồ thị bao gồm các hướng nghiên cứu như : xếp hạng dựa trên nhúng (embedding-based ranking), dự đoán đường đi quan hệ (relation path reasoning), dự đoán dựa trên luật (rule-based reasoning) và học siêu quan hệ. Khai phá thực thể bao gồm nhận dạng, phân biệt, định kiểu và sắp xếp. Các mô hình khai thác quan hệ sử dụng cơ chế chú ý, mạng đồ thị tích chập (graph convolutional networks), huấn luyện đối nghịch (adversarial training), học tăng cường (reinforcement learning), học sâu và học chuyển tiến (transfer learning), đây là hướng nghiên cứu trong phương pháp đề xuất của chúng tôi.

Ngoài ra, trên đồ thị tri thức còn có các hướng nghiên cứu như **đồ thị tri thức về thời gian** và **ứng dụng nhận biết tri thức**. Đồ thị tri thức về thời gian sẽ kết hợp thêm thông tin thời gian trên đồ thị để học cách biểu diễn, còn ứng dụng nhận biết tri thức bao gồm hiểu ngôn ngữ tự nhiên (natural language understanding), trả lời câu hỏi (question answering), hệ thống gợi ý (recommendation systems) và nhiều nhiệm vụ khác trong thế giới thực mà nó tích hợp tri thức vào để cải thiện quá trình học biểu diễn .

Chương 3. PHƯƠNG PHÁP DỰA TRÊN LUẬT

Trong phần này chúng tôi mô tả lại cách mô hình hóa lại bài toán theo phương pháp dựa trên luật AnyBURL, thuật toán lấy mẫu luật (đường dẫn) và thuật toán khái quát hóa một luật để lưu trữ trở thành tri thức của mô hình. Cùng với những cải tiến của chúng tôi trong quá trình đào tạo khi có một tri thức mới được thêm vào đồ thị(thêm cạnh).

3.1 Luật Horn

Trong logic toán học, một công thức nguyên tử (**atomic formula**)[19] còn được gọi đơn giản là một (nguyên tử-**atom**) là một công thức không có cấu trúc mệnh đề, nghĩa là một công thức không chứa các liên kết logic nối liền hoặc nối rời (\vee , \wedge) hoặc tương đương (\Leftrightarrow) là một công thức không có các mẫu con nghiêm ngặt (tức là atom không thể chia nhỏ ra thành các atom con nữa). Do đó, các công thức nguyên tử là công thức đơn giản nhất để hình thành luật của logic. Các công thức hợp được hình thành bằng cách kết hợp các công thức nguyên tử bằng cách sử dụng các liên kết logic.

Một **literal**[6] là một công thức nguyên tử (atom) hoặc phủ định của nó. Định nghĩa chủ yếu xuất hiện trong lý thuyết logic cổ điển. **Literal** có thể được chia thành hai loại: Một **positive literal** chỉ là một nguyên tử (ví dụ: x). Một **negative literal** là phủ định của một nguyên tử (ví dụ: $\neg x$). Sự phân chia của **literal** là **positive literal** hay **negative literal** tùy thuộc vào việc **literal** được định nghĩa.

Một mệnh đề (clause) là một literal hoặc nối rời của hai hoặc nhiều literal. Ở dạng **Horn** một mệnh đề có nhiều nhất một positive literal. Lưu ý: Không phải mọi công thức trong logic mệnh đề đều có thể đưa về dạng Horn. Mệnh đề xác định không có literal đôi khi được gọi là mệnh đề đơn vị (unit clause) và một mệnh đề đơn vị không có biến đôi khi được gọi là *facts*[25]. Một công thức

nguyên tử được gọi là *ground* hoặc *ground atoms* nếu nó được xây dựng hoàn toàn từ các mệnh đề đơn vị; tất cả các *ground atoms* có thể ghép lại từ một tập hợp hàm và các ký hiệu vị từ nhất định tạo nên cơ sở Herbrand cho các bộ ký hiệu này[1].

3.2 Định nghĩa ngôn ngữ đồ thị logic

Khác với các định nghĩa về đồ thị tri thức tổng quát được dùng cho các phương pháp nhúng đồ thị. Trong phương pháp dựa trên luật của chúng tôi. Chúng tôi xem đồ thị như là một ngôn ngữ hình thức. Dưới đây là các định nghĩa theo ngôn ngữ hình thức của đồ thị tri thức.

Một đồ thị tri thức $\mathcal{G}_{\text{know}}$ được định nghĩa trên một bộ từ vựng $\langle \mathbb{C}, \mathbb{R} \rangle$ trong đó \mathbb{C} là tập hợp các hằng số và \mathbb{R} là tập hợp các vị từ nhị phân. Khi đó, $\mathcal{G}_{\text{know}} = \{r(a, b) \mid r \in \mathbb{R}, a, b \in \mathbb{C}\}$ là tập hợp các *ground atoms* hoặc *facts*. Một vị từ nhị phân được gọi là quan hệ và hằng số (hoặc hằng số được đề cập đến) được gọi là thực thể (entity) tương ứng với một dòng dữ liệu trong tập huấn luyện. Sau đây chúng tôi sử dụng các chữ cái viết thường cho các hằng và chữ in hoa cho các biến cho các thảo luận dưới đây. Vì chúng ta không học các quy tắc Horn tùy ý, và chỉ học đối với loại quy tắc nào có thể được khái quát hóa như được thảo luận dưới đây.

Chúng ta định nghĩa một quy tắc là $h(c_0, c_n) \leftarrow b_1(c_0, c_1), \dots, b_n(c_n, c_{n+1})$ là một đường dẫn *ground atoms* có chiều dài n . Trong đó $h(\dots)$ được gọi là *head atoms* và $b_1(c_0, c_1), \dots, b_n(c_n, c_{n+1})$ được gọi là *body atoms*. Chúng tôi sẽ phân biệt dưới đây ba loại quy tắc mà chúng tôi gọi là: *quy tắc nhị phân* (**B**) là quy tắc trong *head atoms* chứa 2 biến, quy tắc đơn nguyên kết thúc bằng một đỉnh treo và *atom* này chỉ chứa biến không chứa hằng số (**U_d**) và *head atoms* chỉ chứa 1 biến. Còn quy tắc đơn nguyên kết thúc bằng một *atom* (**U_c**) và *head atoms* cũng chỉ chứa 1 biến. (**U_c**) có thể là một đỉnh treo tới một hằng số bất kỳ nếu hằng số này trùng với hằng số trong *head atom* thì tạo thành một đường dẫn có chu trình.

$$\begin{aligned}
B : \quad h(A_0, A_n) &\leftarrow \bigwedge_{i=1}^n b_i(A_{i-1}, A_i) \\
U_d : \quad h(A_0, c) &\leftarrow \bigwedge_{i=1}^n b_i(A_{i-1}, A_i) \\
U_c : \quad h(A_0, c) &\leftarrow \bigwedge_{i=1}^{n-1} b_i(A_{i-1}, A_i) \wedge b_n(A_{n-1}, c')
\end{aligned} \tag{3.1}$$

Chúng tôi gọi các quy tắc của các loại này là quy tắc đường đi (path rules), bởi vì các body atoms (phần sau dấu \leftarrow) tạo thành một đường đi. Lưu ý rằng nó cũng bao gồm các biến thể quy tắc với các biến được đảo ngược trong các nguyên tử: được đưa ra trong đồ thị tri thức $\mathcal{G}_{\text{know}}$, đường dẫn có độ dài n là một chuỗi gồm n bộ ba $p_i(c_i, c_i + 1)$ với $p_i(c_i, c_i + 1) \in \mathcal{G}_{\text{know}}$ hoặc $p_i(c_i + 1, c_i) \in \mathcal{G}_{\text{know}}$ với $0 \leq i \leq n$. Các mẫu quy tắc trừu tượng (abstract rule patterns) được cho ở trên có độ dài n vì body atoms của chúng có thể được khởi tạo thành một đường dẫn có độ dài $n - 1$. Ví dụ như Hình 3.1¹. Khi lấy mẫu đường dẫn với độ dài bằng 3 chúng ta có thể có hai quy tắc sau: Quy tắc được đánh dấu màu xanh lá cây và quy tắc được đánh dấu màu đỏ

$$\begin{aligned}
(\text{xanh lá cây}) \quad & \text{speaks}(ed, d) \leftarrow \text{married}(ed, lisa) \wedge \text{born}(lisa, a) \\
(\text{đỏ}) \quad & \text{speaks}(ed, d) \leftarrow \text{lives}(ed, nl) \wedge \text{lang}(nl, d)
\end{aligned}$$

Ngoài ra Quy tắc B và quy tắc U_c cũng được gọi là quy tắc kết nối kín. Chúng được khai thác bởi mô hình AMIE được mô tả trong [13, 14]. Quy tắc U_d là quy tắc không đóng hay đường đi không tạo thành chu trình vì A_n là biến chỉ xuất hiện một lần. Ví dụ:

$$\text{speaks}(X, Y) \leftarrow \text{lives}(X, Y) \tag{1}$$

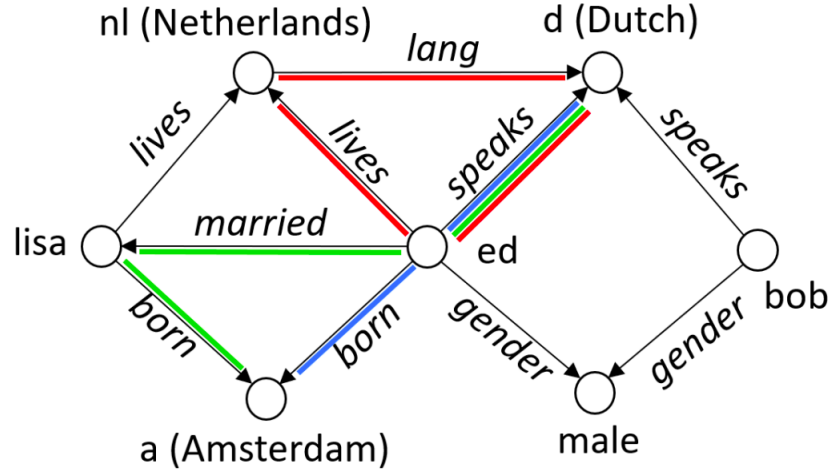
$$\text{lives_in_city}(X, Y) \leftarrow \text{lives}(X, A), \text{within}(Y, A) \tag{2}$$

$$\text{gen}(X, \text{female}) \leftarrow \text{married}(X, A), \text{gen}(A, \text{male}) \tag{3}$$

$$\text{profession}(X, \text{actor}) \leftarrow \text{acted_in}(X, A) \tag{4}$$

Quy tắc (1) là quy tắc **B** (quy tắc nhị phân) quy tắc này nói rằng nếu một người (thực thể) X nói ngôn ngữ Y nếu người X sống ở đất nước Y . Rõ ràng quy tắc này là một quy tắc khái quát miễn khi nào thực thể X có cạnh nối với

¹<http://web.informatik.uni-mannheim.de/AnyBURL/2019-05/meilicke19anyburl.pdf>



Nguồn: tham khảo meilicke19anyburl.pdf

Hình 3.1: Ví dụ đồ thị tri thức

thực thể Y với nhãn là *lives* thì có thể kết thêm 1 cạnh với nhãn *speaks* giữa X và Y . Quy tắc (2), (3) điều là quy tắc U_c , quy tắc (2) nói rằng người X sống ở thành phố Y nếu người X sống ở quốc gia A và thành phố Y nằm trong quốc gia A , quy tắc (3) nói rằng nếu một người X là nữ nếu họ kết hôn với một người A và người A có giới tính nam. Ở quy tắc (3) không tạo thành chu trình trên đồ thị như quy tắc (2) đỉnh (Y) lặp lại ở *head atom* và đỉnh cuối cùng trong *body atoms*. Quy tắc (4) là quy tắc U_d nói rằng người X là một diễn viên nếu người X đóng trong một bộ phim A .

Tất cả các quy tắc được xem xét sẽ được lọc lại dựa trên điểm được gọi là độ tin cậy của quy tắc là được đo trên tập dữ liệu huấn luyện. Độ tin cậy này được đo bằng tỷ lệ *body atoms* dẫn đến *head atoms* chia cho tất cả các đường dẫn chứa *body atoms*. Ví dụ khi ta có quy tắc sau: $gen(X, female) \leftarrow married(X, A), gen(A, male)$. Khi đó chúng ta thực hiện đếm tất cả các cặp thực thể có quan hệ $married(X, A), gen(A, male)$ được gọi là số đường dẫn chứa *body atoms*, sau đó thực hiện đếm tất cả các thực thể thỏa quan hệ $gen(X, female) \leftarrow married(X, A), gen(A, male)$ được gọi là số *body atoms* dẫn đến *head atoms*. Sau đó chia số *body atoms* dẫn đến *head atoms* cho đường dẫn chứa *body atoms* được gọi là độ tin cậy của quy tắc.

3.3 Thuật toán AnyBURL

Trong phần này chúng tôi mô tả lại thuật toán chính của phương pháp AnyBURL nó cũng được mô tả trong [27] cũng như hai thuật toán mở rộng của chúng tôi để giải quyết vấn đề khi đồ thị được thêm một hoặc một lượng tri thức mới (thêm cạnh). Ngoài ra chúng tôi cũng mô tả sơ lược lại cách khởi tạo một luật cũng như cách thức tính toán độ tin cậy bằng cách lấy mẫu trên tập huấn luyện và vấn đề độ tin cậy khi dự đoán một luật khi tính toán độ tin cậy bằng việc lấy mẫu.

3.3.1 AnyBURL

Thuật toán 1 Anytime Bottom-up Rule Learning

```

1: procedure ANYBURL( $\mathcal{G}_{\text{know}}$ ,  $S$ ,  $SAT$ ,  $Q$ ,  $TS$ )
2:    $n = 2$ 
3:    $R = \emptyset$ 
4:   loop
5:      $R_s = \emptyset$ 
6:      $start = currentTime()$ 
7:     repeat
8:        $p = samplePath(\mathcal{G}_{\text{know}}, n)$ 
9:        $R_p = generateRules(p)$ 
10:      for  $r \in R_p$  do
11:         $score(r, s)$ 
12:        if  $Q(r)$  then
13:           $R_s = R_s \cup \{r\}$ 
14:      until  $currentTime() > start + ts$ 
15:       $R'_s = R_s \cap R$ 
16:      if  $|R'_s| / |R| > SAT$  then
17:         $n = n + 1$ 
18:       $R = R_s \cap R$ 
return  $R$ 

```

Đầu vào của thuật toán $\mathcal{G}_{\text{know}}, S, SAT, Q, TS$. Đầu ra là tập hợp R các luật

học được. Trong đó $\mathcal{G}_{\text{know}}$ là một đồ thị tri thức được cho từ tập dữ liệu đào tạo. S là tham số cho biết kích thước của một lần lấy mẫu trên dữ liệu đào tạo để tính toán độ tin cậy. SAT cho biết độ bão hòa(saturation) của các luật được sinh ra trong 1 lần lặp độ bão hòa này được tính bằng số luật **mới** học được ở lần lặp hiện tại so với số luật đã học được. Nếu nhỏ hơn độ bão hòa thì chúng tôi cho rằng vẫn còn tiềm năng để khai thác các luật với độ dài n . Ngược lại chúng tôi tăng độ dài của luật sau đó tiếp tục khai thác. Q là một ngưỡng để xác định xem luật mới được sinh ra có được thêm vào kết quả trả về hay không. Còn TS cho biết thời gian học của thuật toán. Chúng tôi bắt đầu với n bằng 2 tức là các luật có độ dài đường dẫn bằng 2 vì trong path rule yêu cầu ít nhất 1 literal trong head atom và 1 trong body atoms. Ở phần lấy mẫu 1 luật(*samplePath*) chỉ đơn giản là chúng ta chọn một đỉnh bất kì trong đồ thị duyệt qua tất cả các đường dẫn từ đỉnh đó đi qua n đỉnh khác, sau đó chọn ngẫu nhiên một đường dẫn trong số các trường dẫn duyệt được.

3.3.2 Tạo luật

Thuật toán 2 Generate Rules(p)

```

1: procedure GENERATE_RULES(P)
2:   generalizations =  $\emptyset$ 
3:   is_binary_rule = random.choices([true, false])
4:   if is_binary_rule then
5:     replace_all_head_by_variables(p)
6:     replace_all_tail_by_variables(p)
7:     add(generalizations, p)
8:   else:
9:     replace_all_head_by_variables(p)
10:    add(generalizations, p)
11:    replace_all_tail_by_variables(p)
12:    add(generalizations, p)
  return generalizations

```

Ở thuật toán này chúng tôi thay các hằng số vào các head và tail trong toàn bộ path rule của luật được lấy mẫu ở bước trước nếu luật cần học là luật nhị

phân ngược lại chúng tôi chỉ thay hoặc head hoặc tail rồi thêm vào luật trả về sau đó chúng tôi lấy mẫu trên tập huấn luyện 1 tập hợp các luật sau đó tính toán độ tin cậy như được mô tả trong [Tiểu mục 3.3.2](#). Để giảm chi phí tính toán chúng tôi chọn cách lấy mẫu trên tập huấn luyện để tính toán. Khi đưa ra dự đoán các ứng cử viên của một luật chúng tôi sẽ tính toán lại bằng cách thêm vào một lượng biểu diễn số luật bị sai mà chúng tôi chưa nhìn thấy trong quá trình lấy mẫu để tính toán độ tin cậy. Đối với mô hình của chúng tôi sau khi thử nghiệm tham số trong khoảng [5, 10] cho kết quả tốt nhất.

3.4 Thuật toán AnyBURL mở rộng

3.4.1 Thuật toán 3 học offline-to-online

Thuật toán này là phần bổ xung của chúng tôi để tránh việc phải đào tạo lại toàn bộ mô hình khi có một lượng tri thức mới được thêm vào đồ thị. Khi thêm vào đồ thị chúng tôi kiểm tra xem phần tri thức mới có kết nối với tri thức cũ hay không (tính liên thông) nếu có chúng tôi thực hiện phép toán \oplus lấy tất cả các phần trong *batch_edge* thêm với 1 phần liên thông với những cạnh liên thông với đồ thị với độ dài là 5, Nếu không chúng tôi lấy tất cả các phần trong *batch_edge* sau đó thực hiện lại các bước như thuật toán Anytime Bottom-up Rule Learning.

Thuật toán 3 BatchAnyBURL Learning batch size

```
1: procedure BATCHANYBURL( $\mathcal{G}_{\text{know}}$ , SAT, Q, TS, BATCH_EDGE)
2:    $is\_connected = add(\mathcal{G}_{\text{know}}, batch\_edge)$ 
3:   if  $is\_connected$  then
4:      $G' = \mathcal{G}_{\text{know}} \oplus batch\_edge$ 
5:   else
6:      $G' = batch\_edge$ 
7:    $n = 2$ 
8:    $R = \emptyset$ 
9:   loop
10:     $R_s = \emptyset$ 
11:     $start = currentTime()$ 
12:    repeat
13:       $p = samplePath(G', n)$ 
14:       $R_p = generateRules(p)$ 
15:      for  $r \in R_p$  do
16:         $score(r, G')$ 
17:        if  $Q(r)$  then
18:           $R_s = R_s \cup \{r\}$ 
19:    until  $currentTime() > start + ts$ 
20:     $R'_s = R_s \cap R$ 
21:    if  $|R'_s| / |R| > SAT$  then
22:       $n = n + 1$ 
23:     $R = R_s \cap R$ 
return R
```

3.4.2 Thuật toán 4 học online-to-online

Thuật toán 4 EdgeAnyBURL

```
1: procedure EDGEANYBURL( $\mathcal{G}_{\text{KNOW}}$ , S, SAT, Q, TS, EDGE)
2:    $is\_connected = add(\mathcal{G}_{\text{KNOW}}, edge)$ 
3:    $R = \emptyset$ 
4:   if  $is\_connected$  then
5:      $n = 2$ 
6:      $R_s = \emptyset$ 
7:     repeat
8:        $p = samplePath(edge, n)$ 
9:        $R_p = generateRules(p)$ 
10:      for  $r \in R_p$  do
11:         $score(r, s)$ 
12:        if  $Q(r)$  then
13:           $R_s = R_s \cup \{r\}$ 
14:      until  $currentTime() > start + ts$ 
15:       $R'_s = R_s \cap R$ 
16:      if  $|R'| / |R| > SAT$  then
17:         $n = n + 1$ 
18:       $R = R_s \cap R$ 
19:   return R
```

Thuật toán này là một phần bổ xung cho [Thuật toán 3](#). Sở dĩ chúng tôi gọi là online-to-online là vì khi có một cạnh mới(tri thức mới) được thêm vào đồ thị chúng tôi sẽ thực hiện việc học ngay tức khắc trên các path rule liên quan tới cạnh đó không giống như ở [Thuật toán 3](#) khi có đủ 1 lượng tri thức mới được thêm vào.

Chương 4. PHƯƠNG PHÁP DỰA TRÊN HỌC SÂU

Trong phần này chúng tôi trình bày về Đồ Thị Tri Thức (Knowledge Graph), và mô tả lại bài toán Nhúng Đồ Thị (Graph Embedding), sơ lược về các kỹ thuật nhúng đồ thị hiện tại. Chúng tôi sẽ trình bày lại về cơ chế chú ý cũng như cách cơ chế chú ý được áp dụng vào trong đồ thị tri thức bằng mô hình GAT[46]. Đồng thời, chúng tôi sẽ trình bày một phương pháp cải tiến dựa trên mô hình mạng đồ thị chú ý là mô hình KBGAT[32] bằng cách bổ sung thêm thông tin quan hệ và các quan hệ lân cận.

4.1 Nhúng đồ thị

Trong thế giới thực, việc biểu diễn các thực thể và quan hệ thành các vector có thể được hiểu một cách tường minh là quá trình ánh xạ các đặc trưng, các đặc tính của một đối tượng nào đó xuống không gian có số chiều thấp hơn với mỗi thành phần đại diện cho một đặc trưng đơn vị nào đó. Ví dụ, ta biết Donald Trump cao 1m9 và có người vợ là Melania, vì vậy ta có thể biểu diễn thực thể Donald Trump thành một vector

$\vec{e}_{\text{Trump}} = [1.9_{\text{height}}, 0_{\text{area}}, 1_{\text{wife is Melania}}, 0_{\text{wife is Taylor}}]$. Với các đặc trưng không thể đo hoặc không có giá trị (\cdot_{area}) sẽ bằng 0, với các đặc trưng là giá trị mà không có độ lớn (\cdot_{wife}) thì ta chia thành độ lớn là xác suất của các đặc trưng thành phần đơn vị ($\cdot_{\text{wife is Melania}}, \cdot_{\text{wife is Taylor}}$). Như vậy mọi đối tượng trong thế giới thực đều có thể *nhúng* thành các vector một cách tường minh.

Để tìm hiểu về các phương pháp và kỹ thuật *nhúng đồ thị* (graph embedding) cần hiểu các định nghĩa cơ bản như sau :

- **Định nghĩa 5 (Lân Cận Bậc Nhất)** (*First-Order Proximity*) giữa đỉnh v_i và đỉnh v_j là trọng số $A_{i,j}$ của cạnh e_{ij} .

Hai đỉnh giống nhau hơn nếu chúng được kết nối bởi một cạnh có trọng số lớn hơn. Suy ra lân cận bậc nhất giữa đỉnh v_i và v_j là $s_{ij}^{(1)}$, chúng ta có $s_{ij}^{(1)} = A_{i,j}$. Gọi $s_i^{(1)} = [s_{i1}^{(1)}, s_{i2}^{(1)}, \dots, s_{i|V|}^{(1)}]$ biểu thị lân cận bậc nhất giữa v_i và các đỉnh khác. Lấy biểu đồ trong [Hình 2.1](#) làm ví dụ, lân cận bậc nhất v_1 và v_2 là trọng số của cạnh e_{12} , ký hiệu là $s_{12}^{(1)} = 1.2$. Và $s_1^{(1)}$ ghi lại trọng số của các cạnh kết nối v_1 và các đỉnh khác trong đồ thị, tức là, $s_1^{(1)} = [0, 1.2, 1.5, 0, 0, 0, 0, 0, 0]$.

- **Định nghĩa 6 (Lân Cận Bậc Hai)** (*Second-Order Proximity*) $s_{ij}^{(2)}$ ở giữa đỉnh v_i và v_j là sự tương đồng giữa v'_i vùng lân cận $s_i^{(1)}$ và v'_j vùng lân cận $s_j^{(1)}$

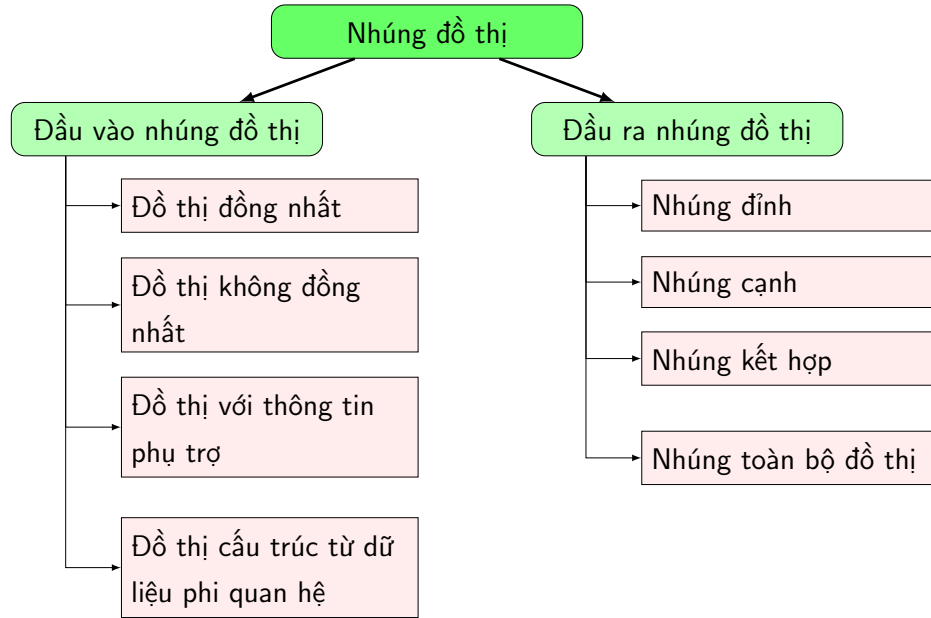
Lấy [Hình 2.1](#) làm ví dụ: $s_{12}^{(2)}$ là điểm tương đồng giữa $s_1^{(1)}$ và $s_2^{(1)}$. Như đã giới thiệu trước, $s_1^{(1)} = [0, 1.2, 1.5, 0, 0, 0, 0, 0, 0]$ và $s_2^{(1)} = [1.2, 0, 0.8, 0, 0, 0, 0, 0, 0]$. Chúng ta hãy xem xét các điểm tương đồng cosine $s_{12}^{(2)} = \text{cosine}(s_1^{(1)}, s_2^{(1)}) = 0.43$ và $s_{15}^{(2)} = \text{cosine}(s_1^{(1)}, s_5^{(1)}) = 0$. Chúng ta có thể thấy rằng lân cận bậc hai giữa v_1 và v_5 bằng 0 vì v_1 và v_5 không chia sẻ bất kỳ hàng xóm 1 hop phổ biến nào. v_1 và v_2 chia sẻ một hàng xóm chung v_3 , do đó khoảng cách thứ hai $s_{12}^{(2)}$ của chúng lớn hơn 0.

Các độ đo lân cận bậc cao hơn (higher-order proximity) có thể được định nghĩa tương tự. Ví dụ, lân cận cách thứ k giữa đỉnh v_i và v_j là sự tương đồng giữa $s_i^{(k1)}$ và $s_j^{(k1)}$.

- **Định nghĩa 7 (Nhúng đồ thị)** Cho đầu vào của đồ thị $\mathcal{G} = (V, E)$ và số chiều được xác định trước của nhúng $d (d \ll |V|)$, vấn đề nhúng đồ thị là chuyển \mathcal{G} thành một không gian d -chiều, trong đó thuộc tính đồ thị được lưu giữ càng nhiều càng tốt. Thuộc tính đồ thị có thể được định lượng bằng cách sử dụng các biện pháp lân cận như lân cận bậc nhất và bậc cao hơn. Mỗi đồ thị được biểu diễn dưới dạng một vector d chiều (cho toàn bộ đồ thị) hoặc một tập các vector d chiều với mỗi vector biểu thị việc nhúng một phần của đồ thị (ví dụ: đỉnh, cạnh, cấu trúc con).

Nhúng đồ thị là quá trình biến đổi các đặc trưng của đồ thị thành các vector hoặc tập hợp những vector có số chiều thấp. Càng nhúng hiệu quả, thì kết quả của độ chính xác trong việc khai thác và phân tích đồ thị sau đó càng cao.

Thách thức lớn nhất của việc nhúng đồ thị phụ thuộc vào cách thiết lập của bài toán (problem setting), bao gồm đầu vào nhúng và đầu ra nhúng như trình bày ở [Hình 4.1](#).



Hình 4.1: Các kỹ thuật nhúng đồ thị

Dựa trên đầu vào nhúng ta phân loại thành các nhóm phương pháp đã khảo sát ở [7] như sau : Đồ thị đồng nhất (homogeneous graph) Đồ thị không đồng nhất (heterogeneous graph) Đồ thị với thông tin phụ trợ (graph with auxiliary information) Đồ thị cấu trúc từ dữ liệu phi-quan hệ (graph constructed from non-relational data).

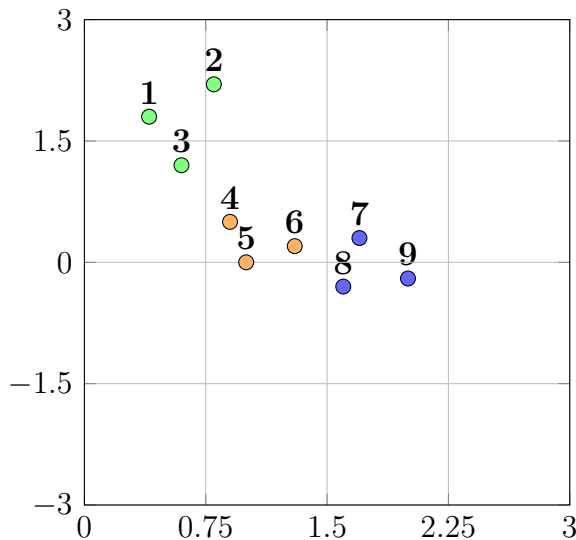
Các loại đầu vào nhúng khác nhau mang thông tin khác nhau được giữ lại trong không gian nhúng và do đó đặt ra những thách thức khác nhau đối với vấn đề nhúng đồ thị. Ví dụ, khi nhúng một đồ thị chỉ với thông tin cấu trúc, các kết nối giữa các đỉnh là mục tiêu cần được lưu giữ. Tuy nhiên, đối với đồ thị có nhãn đỉnh hoặc thông tin thuộc tính của một thực thể, thông tin phụ trợ cung cấp thuộc tính đồ thị từ các ngữ cảnh khác và do đó cũng có thể được xem xét trong quá trình nhúng. Không giống như đầu vào nhúng (embedding input) được cho từ các tập dữ liệu và cố định, đầu ra nhúng (embedding output) được xác định theo từng nhiệm vụ cụ thể. Ví dụ, loại đầu ra nhúng phổ biến nhất là nhúng đỉnh, đại diện cho các đỉnh đóng vai trò như các vector thể hiện độ tương tự giữa các đỉnh. Việc nhúng đỉnh có thể có lợi cho các bài toán liên quan đến đỉnh như phân loại đỉnh, phân cụm đỉnh, v.v.

Tuy nhiên, trong một số trường hợp, các bài toán có thể liên quan đến độ chi tiết cao hơn của đồ thị, ví dụ: cặp đỉnh, đồ thị con, toàn bộ đồ thị. Do đó, thách thức đầu tiên về nhúng là tìm ra loại đầu ra nhúng phù hợp cho ứng dụng quan tâm. Có 4 loại đầu ra nhúng được minh họa ở [Hình 2.1](#) gồm : Nhúng Đỉnh (Node Embedding [4.2](#)), Nhúng Cạnh (Edge Embedding [4.3](#)), Nhúng Kết Hợp (Hybrid Embedding [4.4](#)) và Nhúng Toàn Bộ Đồ Thị (Whole-Graph Embedding [4.5](#)). Các mức độ chi tiết đầu ra khác nhau có các tiêu chí khác nhau sẽ có thách thức khác nhau. Ví dụ, một đỉnh nhúng tốt lưu giữ sự tương tự với các đỉnh lân cận của nó trong không gian nhúng. Ngược lại, việc nhúng toàn bộ đồ thị tốt thể hiện toàn bộ đồ thị dưới dạng một vector sao cho độ tương tự ở mức đồ thị được giữ nguyên.

4.1.1 Phương pháp thiết lập bài toán nhúng đồ thị

Với các đầu vào đã được thiết lập phụ thuộc vào thông tin cần lưu giữ, trong khi đó đầu ra thay đổi tùy theo mục tiêu khai thác đồ thị mà chúng ta mong muốn. Vì vậy ở đây chúng tôi đề cập chi tiết hơn đến các phương pháp thiết lập đồ thị theo kết quả đầu ra trong bài toán nhúng đồ thị.

Nhúng đỉnh

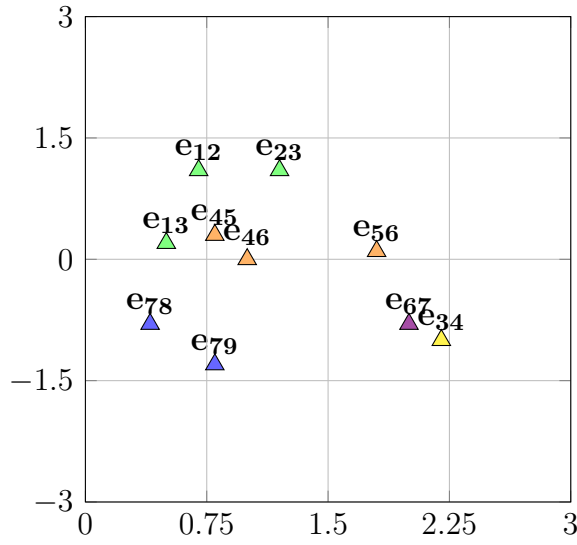


Hình 4.2: Node embedding with each vector representing node features

Nhúng đỉnh (node embedding) biểu diễn mỗi đỉnh như một vector trong không gian số chiều thấp. Các đỉnh *gần* trong đồ thị được nhúng có các biểu

diễn vector tương tự nhau. Sự khác biệt giữa các phương pháp nhúng đồ thị khác nhau nằm ở cách chúng xác định *độ gần nhau* giữa hai đỉnh. Lân cận bậc nhất (Định nghĩa 5) và lân cận bậc hai (Định nghĩa 6)) là hai số liệu thường được sử dụng để tính độ tương tự đỉnh theo cặp. Trong một nghiên cứu, sự gần nhau bậc cao cũng được khám phá ở một mức độ nhất định. Ví dụ nắm bắt các quan hệ hàng xóm k-step ($k = 1, 2, 3, \dots$) trong quá trình nhúng của chúng được đề cập trong nghiên cứu của nhóm tác giả Cao, Shaosheng[8].

Nhúng cạnh



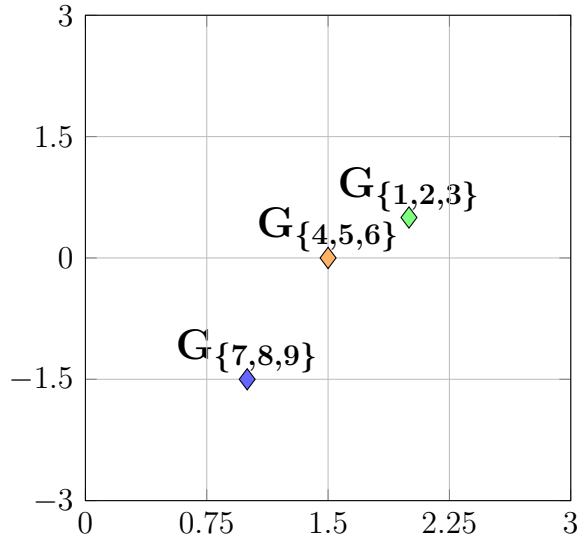
Hình 4.3: Edge embedding with each vector representing edge features

Trái ngược với nhúng đỉnh, nhúng cạnh (edge embedding) nhằm mục đích biểu diễn một cạnh dưới dạng vector có số chiều thấp. Nhúng cạnh hữu ích trong hai trường hợp sau :

Thứ nhất, nhúng đồ thị tri thức. Mỗi cạnh là một bộ ba $\langle h, r, t \rangle$ (Định nghĩa 4). Phép nhúng được học để bảo toàn r giữa h và t trong không gian nhúng, để một thực thể hoặc quan hệ bị thiếu có thể được dự đoán chính xác với hai thành phần còn lại trong $\langle h, r, t \rangle$.

Thứ hai, một số công việc nhúng một cặp đỉnh làm đặc trưng vector để làm cho cặp đỉnh này có thể so sánh với các đỉnh khác hoặc dự đoán sự tồn tại của một liên kết giữa hai đỉnh. Việc nhúng cạnh mang lại lợi ích cho việc phân tích đồ thị liên quan đến cạnh (cặp đỉnh), chẳng hạn như dự đoán liên kết, thực thể biểu diễn tri thức/dự đoán quan hệ, v.v.

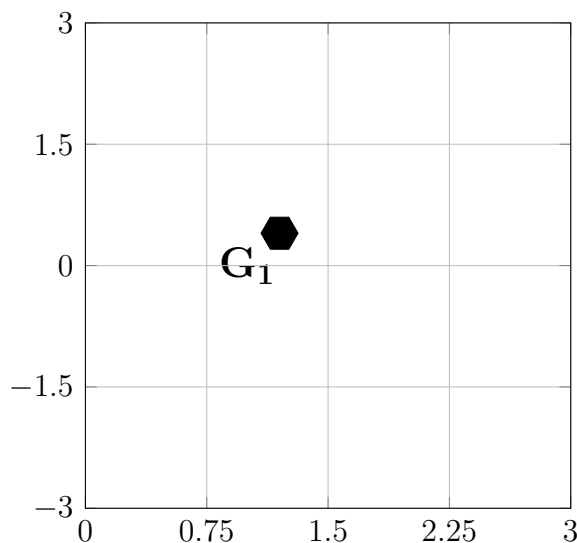
Nhúng Kết Hợp



Hình 4.4: Embedding a graph substructure

Nhúng kết hợp (hybrid embedding) là nhúng kết hợp các loại thành phần đồ thị khác nhau, ví dụ: đỉnh + cạnh (tức là cấu trúc con), đỉnh + bộ phận. Việc nhúng cấu trúc con hoặc bộ phận cũng có thể được bắt nguồn bằng cách tổng hợp các đỉnh riêng lẻ và nhúng cạnh bên trong nó. Tuy nhiên, kiểu tiếp cận *gián tiếp* như vậy không được tối ưu hóa để thể hiện cấu trúc của đồ thị. Hơn nữa, nhúng đỉnh và nhúng bộ phận có thể củng cố lẫn nhau. Nhúng đỉnh tốt hơn vì nó học được cách phối hợp từ sự quan tâm của nhóm lân cận bậc cao, nhúng bộ phận tốt hơn khi phát hiện chính xác hơn đỉnh nhúng được tạo ra.

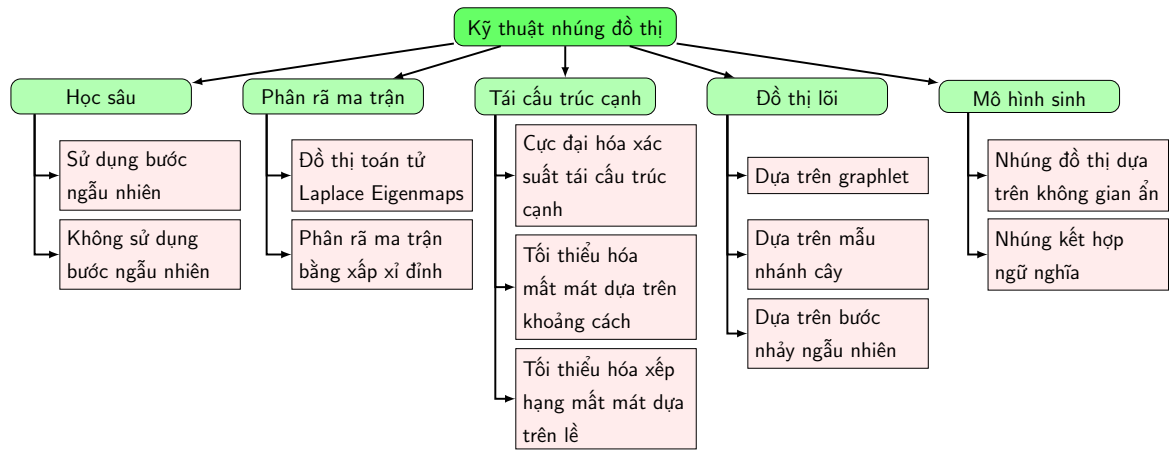
Nhúng Toàn Bộ Đồ Thị



Hình 4.5: Whole-graph embedding

Nhúng toàn bộ đồ thị (whole-graph embedding) thường dành cho các đồ thị nhỏ, chẳng hạn như protein, phân tử, v.v. Trong trường hợp này, một đồ thị được biểu diễn dưới dạng một vector và hai đồ thị tương tự được nhúng để gần nhau hơn. Việc nhúng toàn bộ đồ thị mang lại lợi ích cho nhiệm vụ phân loại đồ thị bằng cách cung cấp một giải pháp đơn giản và hiệu quả để tính toán độ tương đồng của đồ thị. Để thiết lập sự thỏa hiệp giữa thời gian nhúng (tính hiệu quả) và khả năng lưu giữ thông tin (tính biểu đạt), phương pháp Nhúng đồ thị phân cấp [31] thiết kế một khung nhúng đồ thị phân cấp. Nó cho rằng sự hiểu biết chính xác về thông tin đồ thị toàn cục đòi hỏi phải xử lý các cấu trúc con ở các quy mô khác nhau. Một kim tự tháp đồ thị được hình thành trong đó mỗi cấp là một đồ thị tóm tắt ở các tỷ lệ khác nhau. Biểu đồ được nhúng ở tất cả các cấp và sau đó được nối thành một vector. Việc nhúng toàn bộ đồ thị yêu cầu thu thập được thông tin thuộc tính của toàn bộ đồ thị, và vì vậy sẽ tốn nhiều thời gian hơn các phương pháp thiết lập khác.

4.1.2 Các kỹ thuật nhúng đồ thị



Hình 4.6: Các kỹ thuật nhúng đồ thị

Trong phần này, chúng tôi phân loại các nhóm phương pháp nhúng đồ thị dựa vào kỹ thuật sử dụng, như đã nói ở trên, mục tiêu của việc nhúng đồ thị là biểu diễn một đồ thị vào không gian có số chiều thấp mà vẫn giữ vững được những thông tin vốn có của đồ thị nhiều nhất có thể. Các kỹ thuật nhúng đồ thị cơ bản khác nhau ở cách định nghĩa các đặc tính vốn có của đồ thị cần được lưu giữ. Vì mục tiêu chính của chúng tôi là tìm hiểu về các nhóm phương pháp

nhúng đồ thị dựa trên kỹ thuật học sâu nên chúng tôi chỉ trình bày sơ lược đối với các nhóm phương pháp khác.

Học sâu

Ở phần này chúng tôi sẽ trình bày chi tiết về các hướng nghiên cứu của kỹ thuật học sâu (deep learning) bao gồm : sử dụng bước nhảy ngẫu nhiên (random walk) và không sử dụng bước nhảy ngẫu nhiên. Kỹ thuật học sâu được sử dụng phổ biến trong việc nhúng đồ thị bởi vì sự nhanh chóng và hiệu quả trong việc thu thập các đặc trưng một cách tự động. Trong các phương pháp sử dụng kỹ thuật học sâu này, cả 3 loại phương pháp thiết lập đồ thị dựa trên đầu vào (ngoại trừ đồ thị cấu trúc từ dữ liệu phi-quan hệ) và 4 loại đầu ra (Hình 4.1) đều có thể áp dụng kỹ thuật học sâu.

Kỹ thuật học sâu với bước nhảy ngẫu nhiên

Trong nhóm phương pháp này, lân cận bậc hai (Định nghĩa 6) trong đồ thị sẽ được bảo đảm trong không gian nhúng bằng cách cực đại hóa xác suất của những hàng xóm quan sát của một đỉnh điều kiện trên vector nhúng của nó. Đồ thị sẽ được biểu diễn như là một tập hợp mẫu bằng cách lấy mẫu từ những bước đi ngẫu nhiên, và sau đó các phương pháp học sâu sẽ được áp dụng vào đồ thị nhúng để vẫn đảm bảo đặc tính của đồ thị mang theo thông tin đường đi. Các phương pháp sử dụng nhóm phương pháp này như : Deep Walk [35], LINE [40], Node2Vec [17], Anonymous Walk [21], NetGAN [4], ...

Kỹ thuật học sâu không sử dụng bước nhảy ngẫu nhiên

Trong phương pháp này, những cấu trúc học đa lớp sẽ được áp dụng một cách nhanh chóng và hiệu quả để biến đổi đồ thị thành không gian số chiều thấp hơn. Phương pháp này sẽ áp dụng cho toàn bộ đồ thị, có một số phương pháp phổ biến hiện nay đã được khảo sát và trình bày ở báo cáo [37] như sau :

- Mạng Neural Tích Chập (Convolutional Neural Networks)

Mô hình này sử dụng nhiều lớp tích chập : Với mỗi lớp thực hiện tính tích chập trên dữ liệu đầu vào một bộ lọc có số chiều thấp. Kết quả là một ánh xạ đặc trưng, sau đó lại tiếp tục đi qua một lớp kết nối đầy đủ để tính giá trị xác suất. Ví dụ như **ConvE** [11] : Mỗi thực thể và mỗi quan hệ sẽ được biểu diễn bằng một vector số chiều thấp d – chiều. Với mỗi bộ ba, nó ghép và thay đổi kích thước của vector nhúng đỉnh h và quan hệ r vào một đầu vào duy nhất $[h, r]$ với kích thước kết quả là $d_m \times d_n$. Sau đó nó đi

qua lớp tích chập với bộ lọc ω có kích thước $m \times n$, rồi đi qua một lớp kết nối đầy đủ (fully connected layers) và các trọng số W . Kết quả cuối cùng được kết hợp với vector nhúng đuôi t bằng cách sử dụng tích vô hướng (dot products). Kiến trúc này có thể coi là một kiến trúc *phân loại các lớp*.

Một mô hình phổ biến khác là **ConvKB** [33], tương tự như ConvE, nhưng nó ghép cả ba vector nhúng h , r và t vào một ma trận $[h, r, t]$ kích thước $d \times 3$ chiều. Sau đó nó đi qua một lớp tích chập với T bộ lọc ω kích thước 1×3 . Kết quả là $T \times 3$ ánh xạ đặc trưng. Sau đó ánh xạ đặc trưng lại đi qua lớp kết nối đầy đủ và trọng số W . Kiến trúc này có thể coi là kiến trúc phân loại nhị phân.

- Mạng Hồi Quy Tuyến Tính (Recurrent Neural Networks)

Những mô hình này sẽ cho một lớp hoặc nhiều lớp hồi quy tuyến tính để phân tích toàn bộ đường đi (một chuỗi sự kiện/bộ ba) lấy ra từ tập huấn luyện, thay vì chỉ xử lý các sự kiện một cách riêng biệt. Ví dụ như RSN [18], nhận thấy mô hình hồi quy tuyến tính truyền thống không phù hợp cho đồ thị, với mỗi lần thực hiện nó chỉ lấy thông tin của mỗi quan hệ mà không lấy thông tin của vector đỉnh của lần thực hiện trước đó. Vì vậy nó không xử lý rõ ràng sự luân chuyển các đường dẫn của các thực thể và quan hệ. Để giải quyết vấn đề này, họ đề xuất RSN (Recurrent Skipping Networks [18]) : với mỗi bước nhảy, nếu đầu vào là quan hệ, một trạng thái ẩn được cập nhật để tái sử dụng thêm vector đỉnh. Sau đó, kết quả đầu ra được nhân tích vô hướng với mỗi vector nhúng mục tiêu.

- Mạng Neural Bao Bọc (Capsule Neural Networks)

Mạng bao bọc (capsule networks) sẽ sắp một nhóm neural lại với nhau gọi là viên nang, mỗi viên nang này sẽ mã hóa những đặc trưng đặc biệt của đầu vào, như là đại diện cho một nhóm hình ảnh cụ thể. Ưu điểm của mạng bao bọc đó là giúp nhận ra những đặc trưng mà không mất thông tin không gian so với việc tính tích chập thông thường. Mỗi một viên nang tìm ra những đặc trưng theo kích thước vector đầu ra. Ví dụ như : **CapsE** [48], mỗi thực thể và quan hệ được xem là một vector nhúng như trên, tương tự như ConvKB, nó sẽ ghép ba vector nhúng h , r và t thành một ma trận nhúng kích thước $d \times 3$. Sau đó nó đi qua lớp có E bộ lọc tích chập

có kích thước 1×3 . Kết quả là một ma trận kích thước $d \times E$ mà với mỗi dòng thứ i – th đại diện cho những thực thể $h[i]$, $t[i]$ và quan hệ $r[i]$ riêng biệt. Ma trận này sẽ đi lớp bao bọc mà mỗi viên nang (4.1.2) riêng biệt xử lý mỗi cột, vì vậy nó nhận được thông tin dựa theo một đặc trưng của bộ ba đầu vào. Và lớp thứ hai với một lớp bao bọc được sử dụng để đưa ra kết quả đầu ra.

- **Mạng đồ thị chú ý (Graph Attention Networks)**

Nhóm phương pháp này sử dụng cơ chế chú ý (attention mechanism [45]) mà đã đạt được kết quả cải thiện đáng kể trong xử lý ngôn ngữ tự nhiên. Ở nhóm phương pháp này với mỗi vector nhúng, các thực thể được tổng hợp thông tin chú ý từ các thực thể kề cận, sau đó các thông tin chú ý được ghép chồng với nhau và đi qua một lớp kết nối đầy đủ và trọng số để biến đổi thành các vector nhúng cuối cùng. Ví dụ như : GAT [46] với mỗi bộ ba từ tập huấn luyện được nhúng và áp dụng cơ chế chú ý đa đỉnh để cho ra một vector nhúng. Sau đó vector nhúng này tiếp tục đi qua một ma trận trọng số để biến đổi thành vector nhúng mới có số chiều lớn hơn tổng hợp thông tin từ các đỉnh kề cận từ bộ ba ban đầu. Một cải tiến khác của GAT bằng cách thêm thông tin của vector nhúng quan hệ là KBGAT [32]. Các phương pháp này được trình bày cụ thể ở các phần tiếp theo.

- Ngoài ra còn một số phương pháp khác như sử dụng kỹ thuật tự động mã hóa (autoencoder) như Mạng Nhúng Cấu Trúc Sâu (Structural Deep Network Embedding [49]) .

Phân rã ma trận

Phân rã ma trận (matrix factorization) dựa trên đồ thị nhúng biểu diễn những đặc tính của đồ thị (ví dụ những cặp tương đồng hay giống nhau) dưới hình thức một ma trận và phân rã ma trận này để lấy được thông tin nhúng của đỉnh. Đầu vào của nhóm phương pháp này thường là những đặc trưng phi-quan hệ nhiều chiều và đầu ra là tập hợp các đỉnh nhúng. Có hai phương pháp nhúng đồ thị dựa trên phân rã ma trận bao gồm : Đồ Thị Toán Tử Laplace Eigenmaps (Graph Laplacian Eigenmaps) và Phân Rã Ma Trận Xấp Xỉ Đỉnh (Node Proximity Matrix Factorization)

- *Đồ Thị Toán Tử Laplace Eigenmaps*

Nhóm phương pháp này sẽ đảm bảo đặc tính của đồ thị bằng cách phân tích những cặp tương đồng và sẽ phạt nặng những đỉnh có sự tương đồng lớn hơn mà nhúng xa nhau.

- *Phân Rã Ma Trận Xấp Xỉ Đỉnh*

Nhóm phương pháp này sẽ xấp xỉ các đỉnh lân cận trong một không gian số chiều thấp sử dụng kỹ thuật phân rã ma trận. Mục tiêu là để bảo toàn những đỉnh lân cận để tối thiểu hóa hàm xấp xỉ.

Tái cấu trúc cạnh

Phương pháp tái cấu trúc cạnh (edge reconstruction) sẽ xây dựng các cạnh dựa trên những đỉnh nhúng sao cho giống với những đồ thị đầu vào nhất có thể. Phương pháp này tối đa hóa xác suất tái tạo cạnh hoặc tối thiểu hóa hàm mất mát tái tạo cạnh, ngoài ra còn chia ra hàm mất mát dựa trên khoảng cách và hàm xếp hạng mất mát mất mát dựa trên lề .

- *Cực đại hóa xác suất tái cấu trúc cạnh*

Ở phương pháp Cực đại hóa xác suất tái cấu trúc cạnh (maximize edge reconstruct probability), một đỉnh nhúng tốt sẽ cực đại hóa xác suất sinh của các cạnh quan sát trong một đồ thị. Nghĩa là một vector đỉnh nhúng tốt sẽ được tái xây dựng lại như là đồ thị đầu vào gốc. Chúng được phân biệt bằng cách cực đại hóa xuất xuất sinh của tất cả các cạnh quan sát sử dụng vector đỉnh nhúng .

- *Tối thiểu hóa mất mát dựa trên khoảng cách*

Trong phương pháp tối thiểu hóa mất mát dựa trên khoảng cách (minimize distance-based loss), các đỉnh lân cận tính toán dựa trên vector đỉnh nhúng phải càng gần nhất với những đỉnh lân cận trên các cạnh đang quan sát càng tốt. Cụ thể là, độ gần của đỉnh có thể được tính toán dựa trên những đỉnh nhúng hoặc được tính toán theo kinh nghiệm dựa trên các cạnh được quan sát. Sau đó sẽ được tối thiểu hóa sự khác biệt giữa hai loại lân cận để đảm bảo độ gần tương ứng.

- *Tối thiểu hóa xếp hạng mất mát dựa trên lề*

Trong phương pháp tối thiểu xếp hạng mất mát dựa trên lề (minimize margin-based ranking loss), các cạnh của đồ thị đầu vào thể hiện sự tương

quan giữa những cặp đỉnh. Một số đỉnh trong đồ thị thì thường liên kết với những tập hợp đỉnh liên quan. Cụ thể phương pháp này sẽ giúp các đỉnh vector nhúng sẽ gần nhau nếu các đỉnh liên quan đến nhau hơn so với những đỉnh không liên quan khác.

Đồ thị lõi

Với đồ thị lõi (graph kernel) toàn bộ cấu trúc đồ thị có thể được biểu diễn như là một vector chứa số lượng cấu trúc con cơ bản được phân tách từ đồ thị. Kỹ thuật đồ thị lõi bao gồm các nhánh phương pháp con gồm : graphlet, mẫu đồ thị con (subtree patterns) và dựa trên bước nhảy ngẫu nhiên .

Phương pháp này được thiết kế để nhúng toàn bộ đồ thị chỉ lấy đặc trưng toàn cục của toàn bộ đồ thị. Đầu vào của phương pháp này thường là đồ thị đồng nhất. hoặc đồ thị với thông tin bổ trợ

Mô hình sinh

Một mô hình sinh (generative model) có thể được định nghĩa bằng cách xác định sự phân phối chung của đặc trưng đầu vào và những lớp nhãn, và được điều chỉnh dựa trên một tập những tham số. Có hai nhóm phương pháp con của mô hình sinh bao gồm : Nhúng đồ thị dựa trên không gian ẩn (embed graph into latent space) và nhúng kết hợp ngữ nghĩa (incorporate semantics for embedding). Mô hình sinh có thể được dùng cho cả nhúng đỉnh và nhúng cạnh . Nó được xem như là đỉnh những ngữ nghĩa với đầu vào thường là các đồ thị không đồng nhất hoặc đồ thị với thông tin phụ trợ.

- *Nhúng đồ thị trên không gian ngữ nghĩa ẩn*

Với nhóm phương pháp này, các đỉnh được nhúng vào một không gian ngữ nghĩa ẩn nơi khoảng cách giữa đỉnh mô tả được cấu trúc của đồ thị.

- *Nhúng kết hợp ngữ nghĩa*

Phương pháp này thì mỗi đỉnh sẽ gần với đồ thị và có ngữ nghĩa mà nó phải được nhúng gần hơn. Những đỉnh ngữ nghĩa có thể được tìm ra từ những đỉnh mô tả thông qua một mô hình sinh.

Tổng kết : Các phương pháp nhúng đồ thị đều có ưu nhược điểm riêng được nhóm tác giả Cai, Hongyun[7] tổng hợp và trình bày lại ở [Bảng 4.1](#). Với nhóm phương pháp *phân rã ma trận* dựa trên đồ thị nhúng sẽ học những đại diện dựa

trên việc phân tích sự tương đồng các cặp toàn cục. Với nhóm phương pháp *học sâu*, những mô hình này đạt được kết quả hứa hẹn so với những phương pháp khác và phù hợp cho việc nhúng đồ thị vì nó có khả năng học được các biểu diễn phức tạp từ các cấu trúc đồ thị phức tạp.

Bảng 4.1: Bảng so sánh ưu và nhược điểm của kỹ thuật nhúng đồ thị

Phương pháp	Danh mục con	Ưu điểm	Nhược điểm
Phân rã ma trận (Matrix factorization)	Đồ thị toán tử Laplace Eigen-map	Xem xét toàn cục các đỉnh lân cận	Sử dụng không gian và thời gian tính toán lớn
	Phân rã ma trận bằng xấp xỉ đỉnh		
Tái cấu trúc cạnh (Edge reconstruction)	Cực đại hóa xác suất tái cấu trúc cạnh	Huấn luyện tương đối hiệu quả	Tối ưu chỉ sử dụng thông tin cục bộ. Ví dụ như các cạnh (hàng xóm 1 nước) hoặc cặp đỉnh xếp hạng
	Tối thiểu hóa mất mát dựa trên khoảng cách		
	Tối thiểu hóa xếp hạng mất mát dựa trên lề		
Đồ thị lõi (Graph kernel)	Dựa trên graphlet	Hiệu quả, chỉ tính những nhánh cấu trúc đơn vị mong muốn	Nhánh cấu trúc thì không độc lập. Số chiều nhúng tăng lên theo hàm mũ
	Dựa trên mẫu nhánh cây		
	Dựa trên bước nhảy ngẫu nhiên		
Mô hình sinh (Generative model)	Nhúng đồ thị dựa trên không gian ẩn	Phép nhúng có thể giải thích được	Khó điều chỉnh lựa chọn phân bố

Phương pháp	Danh mục con	Ưu điểm	Nhược điểm
	Nhúng kết hợp ngữ nghĩa	Tận dụng nhiều thông tin nguồn	Yêu cầu một lượng lớn dữ liệu huấn luyện một cách tự nhiên
Học sâu (Deep learning)	Sử dụng bước ngẫu nhiên	Hiệu quả và nhanh chóng. Không phải trích đặc trưng	Chỉ xem xét đến nội dung cục bộ trong một đường đi. Khó để tìm kiếm chiến lược lấy mẫu tối ưu
	Không sử dụng bước ngẫu nhiên		Chi phí tính toán cao

Các phương pháp sử dụng kỹ thuật bước nhảy ngẫu nhiên trong học sâu có chi phí tính toán thấp hơn so với các phương pháp sử dụng kỹ thuật học sâu. Các phương pháp truyền thống coi đồ thị như một lưới, tuy nhiên nó không giống với bản chất của đồ thị. Với nhóm phương pháp *tái cấu trúc cạnh* sẽ tối ưu hàm mục tiêu dựa trên các cạnh quan sát hoặc xếp hạng các bộ ba. Nhóm phương pháp này hiệu quả hơn nhưng vector nhúng kết quả lại không quan tâm đến cấu trúc toàn cục của đồ thị. Nhóm phương pháp *đồ thị lõi* chuyển đồ thị vào một vector để dễ dàng thực hiện các nhiệm vụ phân tích đồ thị như phân loại đồ thị. Vì vậy nó chỉ hiệu quả khi liệt kê những nhánh cấu trúc đơn vị mong muốn trong một đồ thị. Với nhóm phương pháp *mô hình sinh*, nó tận dụng thông tin một cách tự nhiên từ nhiều nguồn khác nhau trong một mô hình duy nhất. Việc nhúng đồ thị vào không gian ngữ nghĩa ẩn tạo ra những vector nhúng có thể được diễn giải bằng cách sử dụng ngữ nghĩa. Nhưng giả định về việc lập mô hình quan sát bằng cách sử dụng các phân bố nhất định là khó có thể biện minh. Hơn nữa, phương pháp sinh cần một lượng lớn dữ liệu huấn luyện để ước tính mô hình kết quả phù hợp với dữ liệu. Vì thế nó có thể không đạt kết quả tốt cho những đồ thị nhỏ hoặc số lượng nhỏ đồ thị.

Trong các phương pháp trên, nhóm phương pháp nhúng đồ thị bằng học sâu giúp học được các biểu diễn phức tạp và đạt được kết quả hứa hẹn nhất hiện

nay. Mô hình mạng chú ý trên đồ thị dựa trên cơ chế chú ý giúp tổng hợp thông tin của một thực thể dựa vào các trọng số chú ý của các thực thể lân cận đối với thực thể gốc. Chúng tôi cho rằng đây là hướng nghiên cứu tương tự như quan hệ giữa chú ý và ghi nhớ [47], sự phân bố của chú ý sẽ quyết định trọng số hay sự quan trọng của một thực thể này đối với một thực thể khác. Cũng như vector nhúng biểu diễn cho một thực thể sẽ bị ảnh hưởng bởi sự chú ý hay sự quan trọng của các vector nhúng lân cận. Vì vậy đây là hướng nghiên cứu chúng tôi chọn trong các nhóm phương pháp trên.

4.2 Cơ chế chú ý đa đỉnh

Năm 2014, cơ chế chú ý đa đỉnh (multi-head attention) được phát minh bởi nhóm tác giả Bahdanau, Dzmitry[2] nhưng mãi đến năm 2017 nó mới được phổ biến thông qua mô hình Transformer của nhóm tác giả Vaswani, Ashish[45]. Cơ chế chú ý là một phương pháp hiệu quả giúp thể hiện sự quan trọng của một từ với các từ khác trong một câu, nó còn được chứng minh là đại diện cho bất kỳ phép tính tích chập nào trong báo cáo của nhóm tác giả Cordonnier, Jean-Baptiste[9]. Để hiểu về cách cơ chế chú ý đa đỉnh được áp dụng vào trong đồ thị, trong phần này chúng tôi sẽ trình bày lại chi tiết về cơ chế chú ý đa đỉnh để từ đó hiểu được cách cơ chế chú ý được áp dụng vào nhiệm vụ dự đoán liên kết trong đồ thị tri thức.

4.2.1 Cơ Chế Chú Ý (Attention Mechanism)

Đầu vào của cơ chế chú ý là hai ma trận nhúng $\mathbf{X} = \{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_{N_x}\}$ và $\mathbf{Y} = \{\vec{y}_1, \vec{y}_2, \dots, \vec{y}_{N_y}\}$, với mỗi dòng i^{th} hay j^{th} trong ma trận \mathbf{X} hay \mathbf{Y} là một vector nhúng $\vec{x}_i \in \mathbb{R}^{1 \times D_{\text{in}}}$, $\vec{y}_j \in \mathbb{R}^{1 \times D_{\text{in}}}$. Cơ chế chú ý là quá trình biến đổi vector có D_{in} chiều thành vector đầu ra có $D_{\text{attention}}$ chiều để thể hiện sự quan trọng của từng N_x phần tử x so với tất cả N_y các phần tử y . Với $\mathbf{X} \in \mathbb{R}^{N_x \times D_{\text{in}}}$ và $\mathbf{Y} \in \mathbb{R}^{N_y \times D_{\text{in}}}$ là các ma trận nhúng đầu vào, và $\mathbf{H} \in \mathbb{R}^{N_x \times D_{\text{attention}}}$ là ma trận nhúng đầu ra của cơ chế chú ý của nhóm tác giả Vaswani, Ashish[45] theo công thức sau :

$$\mathbf{H} = \text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V} \quad (4.1)$$

với $\mathbf{Q} = \mathbf{X}\mathbf{W}_Q$, $\mathbf{K} = \mathbf{Y}\mathbf{W}_K$, $\mathbf{V} = \mathbf{Y}\mathbf{W}_V$

Các ma trận trọng số $\mathbf{W}_Q \in \mathbb{R}^{D_{\text{in}} \times D_k}$, $\mathbf{W}_K \in \mathbb{R}^{D_{\text{in}} \times D_k}$ và $\mathbf{W}_V \in \mathbb{R}^{D_{\text{in}} \times D_{\text{attention}}}$ là các ma trận thể hiện quá trình tham số hóa để biến đổi các vector nhúng đầu vào D_{in} chiều thành vector nhúng đầu ra có D_k hoặc $D_{\text{attention}}$ chiều. \mathbf{QK}^T là quá trình nhân tích vô hướng của từng vector nhúng x ban đầu với tất cả vector nhúng y . Việc chia cho $\sqrt{d_k}$ là để chuẩn hóa theo số chiều k . Sau đó kết quả được chuẩn hóa lại bằng hàm *softmax* để có thể so sánh giữa các hệ số chú ý khác nhau. Ta có thể coi $\text{softmax}\left(\frac{\mathbf{QK}^T}{\sqrt{d_k}}\right)$ là một *hệ số chú ý* thể hiện sự quan trọng của từng phần tử y đối với mỗi phần tử x . Cuối cùng, kết quả được nhân với vector nhúng \mathbf{V} để biến đổi từ vector nhúng D_k chiều thành vector nhúng mới $D_{\text{attention}}$ chiều.

Nếu $\mathbf{X} = \mathbf{Y}$ nghĩa là chúng ta đang tính sự quan trọng của một phần tử so với chính các phần tử khác trong ma trận nhúng ban đầu và ta gọi nó là cơ chế tự-chú ý (self-attention mechanism).

4.2.2 Chú Ý Đa Đầu (Multi-Head Attention)

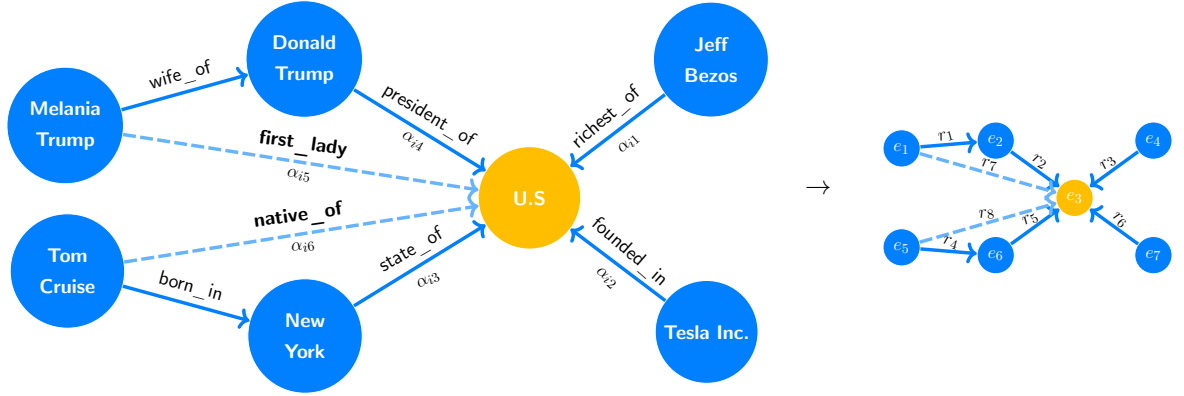
Cơ chế chú ý đa đầu là cách ghép các lớp chú ý ở trên lại để giúp ổn định quá trình học. Tương tự như cơ chế chú ý ở trên, cơ chế chú ý đa đầu (multi-head attention mechanism) là quá trình biến đổi N_x vector nhúng ban đầu D_{in} chiều thành vector nhúng $D_{\text{multi-head}}$ chiều với thông tin được tổng hợp từ nhiều đầu khác nhau giúp ổn định trong quá trình huấn luyện. Cơ chế chú ý đa đầu sẽ ghép N_{head} các đầu ma trận chú ý \mathbf{H} rồi sau đó tiếp tục nhân với một ma trận trọng số để biến đổi từ ma trận nhúng $\mathbf{X} \in \mathbb{R}^{N_x \times D_{\text{in}}}$ ban đầu thành ma trận nhúng mới $\mathbf{X}' \in \mathbb{R}^{N_x \times D_{\text{multi-head}}}$ như công thức sau :

$$\begin{aligned} \mathbf{X}' &= \left(\bigparallel_{h=1}^{N_{\text{head}}} \mathbf{H}^{(h)} \right) \mathbf{W}^O \\ &= \left(\bigparallel_{h=1}^{N_{\text{head}}} \text{Attention}(\mathbf{XW}_Q^{(h)}, \mathbf{YW}_K^{(h)}, \mathbf{YW}_V^{(h)}) \right) \mathbf{W}^O \end{aligned} \quad (4.2)$$

Trong đó các ma trận trọng số $\mathbf{W}_Q^{(h)}$, $\mathbf{W}_K^{(h)} \in \mathbb{R}^{D_{\text{in}} \times D_k}$ và $\mathbf{W}_V^{(h)} \in \mathbb{R}^{D_{\text{in}} \times D_{\text{attention}}}$ thuộc vào từng lớp chú ý $h \in [N_{\text{head}}]$ khác nhau. $\mathbf{W}^O \in \mathbb{R}^{N_{\text{head}} D_{\text{attention}} \times D_{\text{multi-head}}}$ để tham số hóa quá trình biến đổi ma trận các đầu đã ghép thành một ma trận nhúng kết quả cuối cùng.

Đến đây chúng tôi đã trình bày về cơ chế chú ý tính hệ số chú ý và tổng hợp thông tin nhúng từ các vector nhúng lân cận. Trong phần tiếp theo chúng tôi sẽ trình bày về cách cơ chế chú ý được áp dụng vào trong đồ thị tri thức.

4.3 Mạng đồ thị chú ý



Hình 4.7: Đồ thị tri thức và các hệ số chú ý chuẩn hóa của thực thể

Với thành công của *cơ chế chú ý đa đỉnh* trong ngôn ngữ tự nhiên, nó còn được nghiên cứu để áp dụng vào các mô hình của xử lý ảnh [36]. Chính vì vậy cơ chế chú ý đa đỉnh đã được nghiên cứu để áp dụng vào các mô hình nhúng đồ thị tri thức thay cho phương pháp tính chập như Mạng Đồ Thị Tích Chập (GCNs [23]). Ở phần này chúng tôi sẽ trình bày chi tiết về cách cơ chế chú ý ở 4.2.1 được áp dụng vào việc nhúng đồ thị theo phương pháp Mạng Đồ Thị Chú Ý (Graph Attention Network - GAT [46]).

Đầu vào của mô hình *mạng đồ thị chú ý* là tập hợp các vector nhúng được khởi tạo ngẫu nhiên theo phân phối chuẩn biểu diễn đặc trưng của từng thực thể (entity) : $\mathbf{E} = \{\vec{e}_1, \vec{e}_2, \dots, \vec{e}_{N_e}\}$. Mục tiêu của mô hình là biến đổi thành ma trận nhúng đầu ra mới $\mathbf{E}'' = \{\vec{e}_1'', \vec{e}_2'', \dots, \vec{e}_{N_e}''\}$ với khả năng tổng hợp thông tin nhúng từ các thực thể lân cận; $\mathbf{E} \in \mathbb{R}^{N_e \times D_{in}}$ và $\mathbf{E}'' \in \mathbb{R}^{N_e \times D''}$ tương ứng là ma trận nhúng đầu vào và ma trận nhúng đầu ra của tập hợp thực thể, N_e là kích thước của tập thực thể, D_{in} và D'' tương ứng là số chiều nhúng đầu vào, và số chiều nhúng đầu ra.

Tương tự như cơ chế chú ý đa đỉnh được trình bày ở mục 4.2.1, việc áp dụng của cơ chế chú ý đa đỉnh trên đồ thị tri thức sẽ áp dụng với chính mỗi vector nhúng thực thể giống như *cơ chế tự-chú ý* (self-attention mechanism), mỗi đỉnh

sẽ chú ý với tất cả các đỉnh khác trong đồ thị. Tuy nhiên, việc tính hệ số chú ý giữa tất cả các đỉnh với nhau trong đồ thị là không có ý nghĩa nếu không có mối quan giữa chúng và khối lượng tính toán rất lớn, vì vậy mô hình áp dụng cơ chế gọi là *mặt nạ chú ý* (mask attention) bằng cách bỏ đi tất cả những hệ số chú ý không có quan hệ trong đồ thị, đó chính xác là giá trị của lân cận bậc nhất (Định nghĩa 5) của một đỉnh trong đồ thị. Khi đó $\mathbf{X} = \mathbf{Y} = \mathbf{E}$ (4.2.1) và hệ số chú ý của của cơ chế mặt nạ chú ý được hiểu là sự quan trọng của một đỉnh $j \in \mathcal{N}_i$ đối với đỉnh gốc i , với \mathcal{N}_i là tập hợp tất cả những hàng xóm của đỉnh i (bao gồm cả i).

Việc áp dụng cơ chế chú ý đa đỉnh (*multi-head attention*) ở 4.2 vào đồ thị được mô tả như sau :

$$e_{ij} = f_{\text{mask attention}}(\mathbf{W}\vec{e}_i, \mathbf{W}\vec{e}_j) \quad (4.3)$$

trong đó e_{ij} là hệ số chú ý đa đỉnh của một cạnh (e_i, e_j) đối với thực thể gốc e_i trong đồ thị $\mathcal{G}_{\text{know}}$. \mathbf{W} là ma trận trọng số để tham số hóa quá trình biến đổi tuyến tính. $f_{\text{mask attention}}$ là hàm áp dụng cơ chế chú ý.

Trong mô hình GAT, mô hình sẽ đi qua hai quá trình biến đổi vector nhúng \vec{e}_i của thực thể e_i . Toàn bộ mô hình bao gồm hai bước biến đổi, với mỗi bước là một quá trình biến đổi vector nhúng bằng cơ chế chú ý đa đỉnh như sau :

$$\vec{e}_i \xrightarrow{f_{\text{mask attention}}^{(1)}} \vec{e}_i' \xrightarrow{f_{\text{mask attention}}^{(2)}} \vec{e}_i'' \quad (4.4)$$

Ở quá trình chú ý đa đỉnh đầu tiên ($f_{\text{mask attention}}^{(1)}$), mô hình sẽ tổng hợp thông tin từ các thực thể lân cận và ghép chồng lên nhau để tạo ra vector \vec{e}_i' , với $\vec{e}_i' \in \mathbb{R}^{1 \times D'}$. Ở bước thứ hai ($f_{\text{mask attention}}^{(2)}$), lớp chú ý đa đỉnh đã định không còn nhạy cảm với quá trình tự-chú ý nên kết quả sẽ được tính *trung bình* thay vì ghép các đỉnh chú ý lại với nhau, vector \vec{e}_i' tiếp tục được xem là vector nhúng đầu vào để biến đổi thành vector nhúng \vec{e}_i'' cuối cùng với $\vec{e}_i'' \in \mathbb{R}^{1 \times D''}$.

Đầu tiên, giống như cơ chế chú ý 4.1, mỗi vector nhúng sẽ được nhân với một ma trận trọng số $\mathbf{W}_1 \in \mathbb{R}^{D_k \times D_{\text{in}}}$ để thể tham số hóa quá trình biến đổi tuyến tính từng vector nhúng của thực thể từ số chiều D_{in} lên số chiều D_k có đặc trưng cao hơn :

$$\vec{h}_i = \mathbf{W}_1 \vec{e}_i \quad (4.5)$$

khi đó $\vec{e}_i \in \mathbb{R}^{D_{in} \times 1} \rightarrow \vec{h}_i \in \mathbb{R}^{D_k \times 1}$

Sau đó, ta ghép các cặp vector nhúng thực thể vừa biến đổi tuyến tính với nhau để tính hệ số chú ý, hệ số chú ý e_{ij} thể hiện sự quan trọng của đặc trưng cạnh (e_i, e_j) đối với thực thể gốc e_i hay sự quan trọng của một thực thể e_j có quan hệ với thực thể gốc e_i , ta áp dụng hàm LeakyReLU để lấy giá trị tuyệt đối của hệ số chú ý, mỗi hệ số chú ý e_{ij} được tính theo công thức sau :

$$e_{ij} = \left(\text{LeakyReLU} \left(\vec{\mathbf{W}}_2^T [\vec{h}_i || \vec{h}_j] \right) \right) \quad (4.6)$$

với $.^T$ là phép chuyển vị, $||$ là phép ghép. Tương tự 4.1, tuy nhiên thay vì thực hiện tính tích vô hướng thì ta sử dụng một *cơ chế chú ý chung* (shared attentional mechanism) $\vec{\mathbf{W}}_2 : \mathbb{R}^{D_k} \times \mathbb{R}^{D_k} \rightarrow \mathbb{R}$ để tính hệ số chú ý. Như đã trình bày ở 4.3, ta thực hiện tự chú ý giữa tất cả các đỉnh với nhau bằng cơ chế mặt nạ chú ý để bỏ hết tất cả thông tin cấu trúc. Để có thể dễ dàng so sánh các hệ số chú ý với nhau giữa tất cả các thực thể, một hàm *softmax* được áp dụng để chuẩn hóa trên tất cả các hàng xóm e_j có quan hệ với thực thể gốc e_i : $\alpha_{ij} = \text{softmax}_j(e_{ij})$. Kết hợp lại ta có công thức của hệ số chú ý chuẩn hóa của từng hàng xóm đối với thực thể gốc như sau :

$$\alpha_{ij} = \frac{\exp \left(\text{LeakyReLU} \left(\vec{\mathbf{W}}_2^T [\vec{h}_i || \vec{h}_j] \right) \right)}{\sum_{k \in \mathcal{N}_i} \exp \left(\text{LeakyReLU} \left(\vec{\mathbf{W}}_2^T [\vec{h}_i || \vec{h}_k] \right) \right)} \quad (4.7)$$

Ở bước này, mô hình GAT tương tự như GCN [23], các vector nhúng từ hàng xóm sẽ được tổng hợp với nhau và mở rộng hay thu nhỏ (scale) theo hệ số chú ý đã chuẩn hóa :

$$\vec{e}_i = \sigma \left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \vec{h}_j \right) \quad (4.8)$$

Tương tự như lớp chú ý đa đỉnh, ta sẽ ghép N_{head} đỉnh lại với nhau để giúp ổn định quá trình học ở bước ($f_{\text{mask attention}}^{(1)}$ 4.4) đầu tiên của mô hình:

$$\vec{e}_i = \prod_{h=1}^{N_{\text{head}}} \sigma \left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^h \mathbf{W}^h \vec{h}_j \right) \quad (4.9)$$

trong đó σ là bất kỳ hàm biến đổi phi tuyến tính nào, α_{ij}^h là hệ số chú ý được chuẩn hóa của cạnh (e_i, e_j) được tính từ lớp thứ h^{th} , tương tự như công thức 4.1 \mathbf{W}^h là ma trận trọng số để biến đổi tuyến tính vector nhúng đầu vào, với \mathbf{W}^h thuộc các lớp ghép chồng h^{th} khác nhau. Cuối cùng vector nhúng mới $\vec{e}_i' \in \mathbb{R}^{1 \times D'}$ với $D' = N_{\text{head}} D_k$ tiếp tục được xem là vector đầu vào để thực hiện cơ chế chú ý. Tuy nhiên ở bước thứ hai ($f_{\text{mask attention}}^{(2)}$ 4.4) giá trị chú ý đa đỉnh sẽ được tính trung bình thay vì ghép chồng lên nhau theo công thức sau :

$$\vec{e}_i'' = \sigma \left(\frac{1}{N_{\text{head}}} \sum_{h=1}^{N_{\text{head}}} \sum_{j \in \mathcal{N}_i} \alpha_{ij}^h \mathbf{W}^h \vec{e}_j \right) \quad (4.10)$$

Tổng kết : Đến đây chúng tôi đã trình bày được cách cơ chế chú ý tổng hợp một vector nhúng thực thể trong đồ thị tri thức từ các vector nhúng lân cận và ghép lại với nhau để tạo ra vector nhúng kết quả. Trong phần tiếp theo chúng tôi sẽ trình bày đầy đủ về mô hình nhúng của chúng tôi dựa trên mô hình KBGAT của nhóm tác giả Nathani, Deepak[32] .

4.4 Mô hình KBGAT

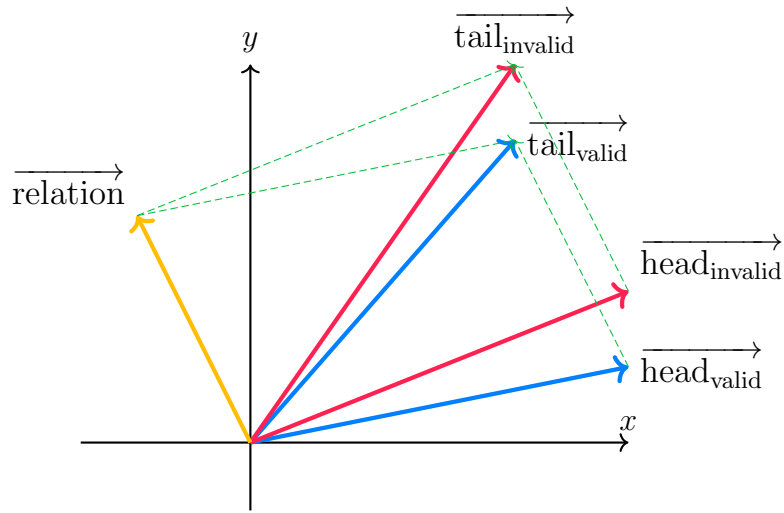
Trong đồ thị tri thức, một thực thể không thể là một đại diện đầy đủ cho một cạnh, vì một thực thể có thể đóng nhiều vai trò khác nhau phụ thuộc vào từng loại quan hệ khác nhau. Ví dụ như Hình 4.7, Donald Trump vừa đóng là vai trò là tổng thống, vừa đóng vai trò là người chồng. Để giải quyết vấn đề này, mô hình nhúng đồ thị dựa trên chú ý - KBGAT (graph attention based embeddings [32]) là một cải tiến của mô hình GAT bằng cách kết hợp thêm thông tin của *quan hệ và đặc trưng các đỉnh hàng xóm* vào trong cơ chế chú ý. Trong phần này chúng tôi sẽ trình bày chi tiết về mô hình KBGAT. Cấu trúc của KBGAT là một mô hình mã hóa-giải mã (encoder-decoder) trong đó lớp mã hóa chúng tôi sử dụng mô hình Mạng đồ thị chú ý, còn với mô hình giải mã chúng tôi sử dụng mô hình ConvKB để tiến hành dự đoán. Các bước của mô hình KBGAT được minh họa ở 4.11.

$$entities \xrightarrow{\text{Embedding}^{4.4.1}} e_{\text{TransE}} \xrightarrow{\text{Embedding}^{4.4.2}} e_{\text{KBGAT}} \xrightarrow{\text{ConvKB}^{4.4.3}} e_{\text{prob}} \quad (4.11)$$

Đầu tiên, các vector nhúng của mỗi thực thể được khởi tạo bằng mô hình

TransE để học được các đặc trưng không gian giữa các đỉnh để thu được vector nhúng. Sau đó các vector nhúng này được tiếp tục huấn luyện bằng mô hình mã hóa để học được các đặc trưng lân cận để có được vector nhúng mới. Cuối cùng, các vector nhúng này đi qua lớp dự đoán bằng mô hình ConvKB. Tất cả các công thức của chúng tôi đều được trình bày lại theo nhóm tác giả Nathani, Deepak[32].

4.4.1 Khởi tạo vector nhúng



Hình 4.8: Minh họa về các vector nhúng trong mô hình TransE

Tương tự như phương pháp Word2Vec[30], mô hình *Biến đổi vector nhúng để mô hình hóa dữ liệu đa-quan hệ* (TransE [5]) thuộc nhóm các phương pháp nhúng hình học để biến đổi các thực thể và quan hệ trong đồ thị tri thức thành các vector nhúng đầu ra sao cho :

$$\overrightarrow{\text{entity}_{\text{head}}} + \overrightarrow{\text{relation}} \approx \overrightarrow{\text{entity}_{\text{tail}}} \quad (4.12)$$

Đầu tiên, các vector nhúng thực thể và quan hệ được khởi tạo ngẫu nhiên bằng phân phối chuẩn theo số chiều của vector khởi tạo D_{in} , sau đó được chuẩn hóa theo kích thước của tập thực thể nhúng, và quan hệ nhúng. Sau đó, ta lấy mẫu (sampling) từ tập dữ liệu huấn luyện để có được một *lô bộ ba hợp lệ* (S_{batch}). Với mỗi bộ ba như vậy, chúng ta lấy mẫu bộ ba không hợp lệ bằng cách thay thực thể đầu hoặc thực thể đuôi bằng một thực thể ngẫu nhiên trong tập thực thể để được *lô bộ ba không hợp lệ* (S'_{batch}). Sau đó ta nhóm từng bộ ba hợp

lệ với không hợp lệ để tạo ra *lô huấn luyện* (T_{batch}). Cuối cùng, ta cập nhật giá trị các vector nhúng để đảm bảo điều kiện 4.12 .

Thuật toán 5 Thuật toán học vector nhúng TransE [5]

Input : Tập huấn luyện $S = (h, r, t)$, tập thực thể E , tập quan hệ R , biên lề γ , số chiều nhúng D_{in}

Initialize

- 1: $\vec{r} \leftarrow \text{uniform}(-\frac{6}{\sqrt{D_{\text{in}}}}, \frac{6}{\sqrt{D_{\text{in}}}})$ với mỗi quan hệ $r \in R$
- 2: $\vec{r} \leftarrow \frac{\vec{r}}{\|\vec{r}\|}$ với mỗi $r \in R$
- 3: $\vec{e} \leftarrow \text{uniform}(-\frac{6}{\sqrt{D_{\text{in}}}}, \frac{6}{\sqrt{D_{\text{in}}}})$ với mỗi thực thể $e \in E$
- 4: **loop**
- 5: $\vec{e} \leftarrow \frac{\vec{e}}{\|\vec{e}\|}$ với mỗi $e \in E$
- 6: $S_{\text{batch}} \leftarrow \text{sample}(S, b)$ // lấy mẫu minibatch kích thước b
- 7: $T_{\text{batch}} \leftarrow \emptyset$
- 8: **for** $(h, r, t) \in S_{\text{batch}}$ **do**
- 9: $(h', r, t') \leftarrow \text{sample}(S'_{(h,r,t)})$ // lấy mẫu từ bộ ba không hợp lệ
- 10: $T_{\text{batch}} \leftarrow T_{\text{batch}} \cup \left\{ \left((h, r, t), (h', r, t') \right) \right\}$

Cập nhật nhúng

$$11: \quad \sum_{(h,r,t),(h',r,t') \in T_{\text{batch}}} \nabla[\gamma + d(\vec{h} + \vec{r}, \vec{t}) - d(\vec{h}' + \vec{r}, \vec{t})]_+$$

Output : tập các vector có số chiều là D_{in} đại diện cho các thực thể và quan hệ

Mô hình TransE của nhóm tác giả Bordes, Antoine[5] được trình bày ở thuật toán 5.

Trong đó, đầu vào của mô hình TransE là tập dữ liệu huấn luyện với mỗi phần tử là một bộ ba (h, r, t) . $h, t \in E$ là các thực thể đầu và thực thể đuôi, $r \in R$ là các quan hệ. \vec{e} và \vec{r} lần lượt là các vector nhúng của thực thể và quan hệ, và $\|\vec{e}\|$ và $\|\vec{r}\|$ lần lượt là độ lớn của tập thực thể và tập quan hệ. S và S_{batch} tương ứng là tập dữ liệu huấn luyện, và một lô (batch) lấy ra từ tập dữ liệu huấn luyện. T_{batch} là một lô bao gồm cả bộ ba hợp lệ và bộ ba không hợp lệ để tính hàm mất mát 4.13 .

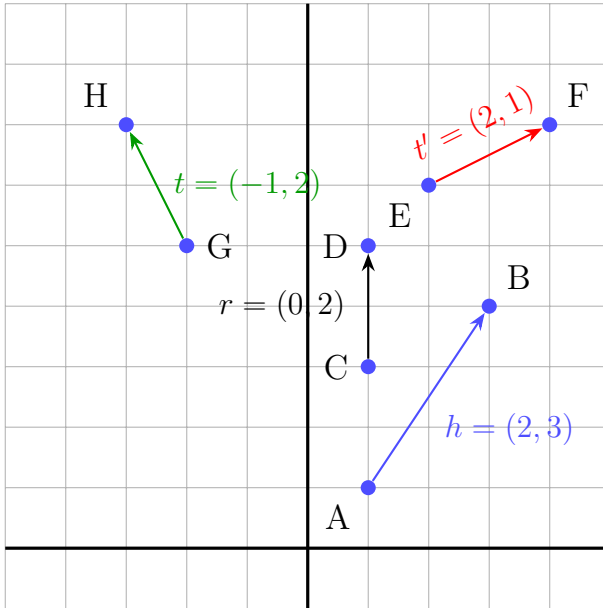
Bộ ba hợp lệ (valid triple) là bộ ba lấy từ lô huấn luyện (S_{batch}), *bộ ba không hợp lệ* (invalid triple) là bộ ba lấy từ lô huấn luyện lỗi (S'_{batch}) được bỏ đi thực

thể đầu hoặc thực thể đuôi và thay thế các thực thể đó bằng cách chọn một thực thể ngẫu nhiên trong tập thực thể :

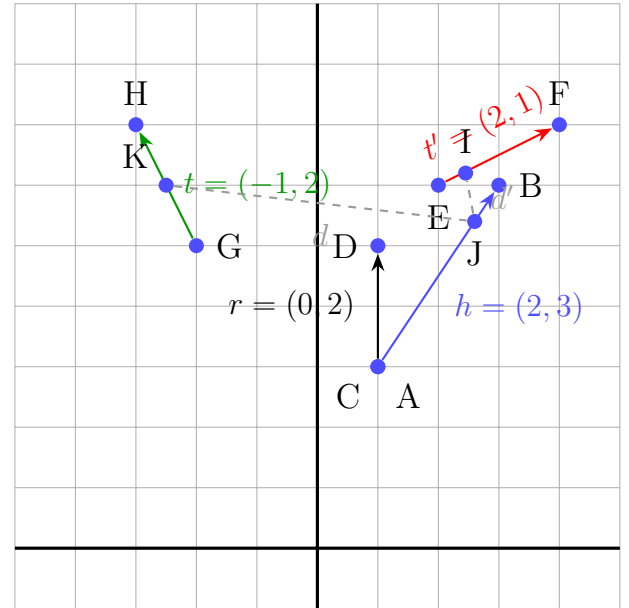
$$S'_{(h,r,t)} = \{(h', r, t) | h' \in E\} \cup \{(h, r, t') | t' \in E\} \quad (4.13)$$

Để đạt được mục tiêu là tạo ra các vector nhúng sao cho $\vec{h} + \vec{r} \approx \vec{t}$, ý tưởng của mô hình là các vector nhúng trong lô hợp lệ thì vector nhúng đuôi \vec{t} phải ở lân cận của $\vec{h} + \vec{r}$, ngược lại các vector nhúng trong lô không hợp lệ $\vec{h} + \vec{r}$ (hoặc \vec{t}) thì phải nằm xa so với \vec{t} (hoặc $\vec{h} + \vec{r}$) theo hàm mất mát sau :

$$\mathcal{L} = \sum_{(h,r,t) \in S} \sum_{(h',r,t') \in S'_{(h,r,t)}} [d - d' + \gamma]_+ \quad (4.14)$$



(a) Vector embedding TransE ban đầu



(b) TransE embedding sau khi biến đổi

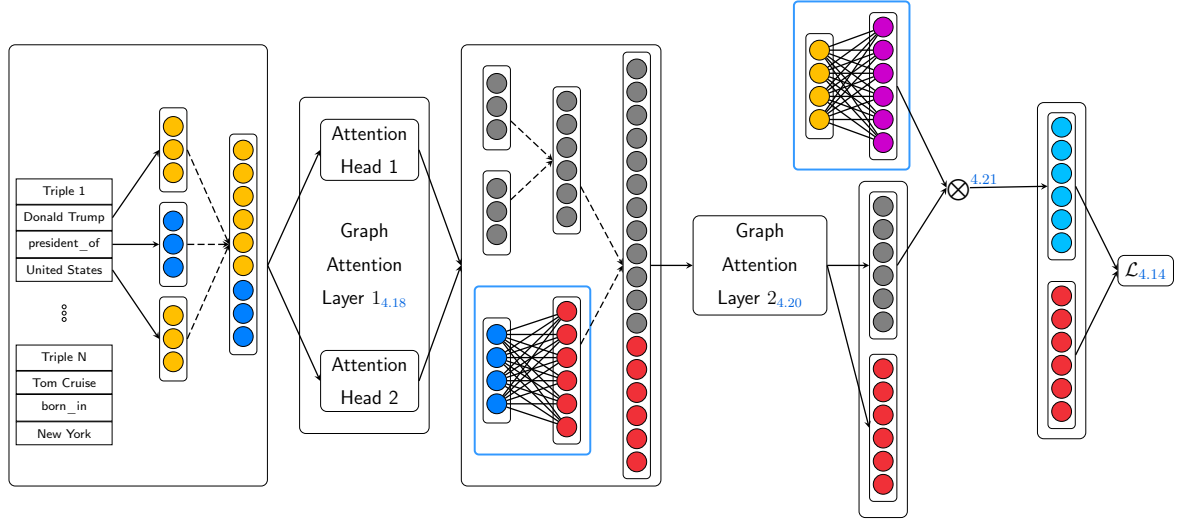
Hình 4.9: Mô hình nhúng đồ thị TransE

Với $\gamma > 0$ là biên lề, h' và t' lần lượt là các thực thể lấy mẫu trong công thức 4.13 ; Δ và Δ' lần lượt là hàm đo giá trị sai khác của các thực thể nhúng trong lô hợp lệ $d = \|\Delta\|_1 = \|\vec{h} + \vec{r} - \vec{t}\|_1$ và lô không hợp lệ $d' = \|\Delta'\|_1 = \|\vec{h} + \vec{r} - \vec{t}'\|_1$ ($\|\cdot\|_1$ là hàm chuẩn hóa L1-norm) . Theo minh họa trong Hình 4.9, ta thấy nếu $d > d'$ hay $d - d' > 0$ thì $\vec{h} + \vec{r}$ sẽ nằm gần \vec{t} hơn so với \vec{t}' . Ta muốn các vector nhúng thỏa mãn điều kiện 4.12 nên $\vec{h} + \vec{r}$ phải càng gần t càng tốt, nghĩa là

$\vec{h} + \vec{r}$ càng gần \vec{t} thì càng không đúng. Vì vậy trong quá trình huấn luyện ta muốn Δ' càng lớn bằng Δ càng tốt và nếu $\Delta' > \Delta$ hay $d' - d > 0$ thì ta không cần cập nhật các trọng số nhúng nữa. Vì vậy trong hàm mất mát 4.14 $[d - d' + \gamma]_+$ là ta muốn lấy phần dương vì phần âm đã đảm bảo tính đúng đắn của điều kiện 4.12 trong quá trình huấn luyện.

4.4.2 Mô hình mã hóa

Sau khi có được các vector nhúng học được các đặc trưng không gian ở trên của đồ thị. Các vector nhúng tiếp tục được đi qua lớp nhúng tiếp theo để tổng hợp thêm thông tin lân cận của một thực thể.



Hình 4.10: Minh họa các lớp mã hóa của mô hình KBGAT

Mô hình sẽ biến đổi từ ma trận nhúng thực thể $\mathbf{E} = \{\vec{e}_1, \vec{e}_2, \dots, \vec{e}_{N_e}\} \rightarrow \mathbf{E}'' = \{\vec{e}_1'', \vec{e}_2'', \dots, \vec{e}_{N_e}''\}$, với $\mathbf{E} \in \mathbb{R}^{N_e \times D_{in}}$ và $\mathbf{E}'' \in \mathbb{R}^{N_e \times D''}$. Đồng thời biến đổi ma trận nhúng quan hệ $\mathbf{R} = \{\vec{r}_1, \vec{r}_2, \dots, \vec{r}_{N_r}\} \rightarrow \mathbf{R}'' = \{\vec{r}_1'', \vec{r}_2'', \dots, \vec{r}_{N_r}''\}$ với $\mathbf{R} \in \mathbb{R}^{N_r \times P_{in}}$ và $\mathbf{R}'' \in \mathbb{R}^{N_r \times P''}$. Tương tự như mô hình GAT đã trình bày ở mục 4.3, mô hình sẽ biến đổi vector nhúng thực thể D_{in} chiều thành D'' chiều với thông tin được tổng hợp từ các hệ số chú ý lân cận. P_{in} và P'' lần lượt là số chiều của vector nhúng quan hệ đầu vào và đầu ra. N_e , N_r tương ứng là kích thước của tập thực thể và tập quan hệ trong \mathcal{G}_{know} .

Mô hình KBGAT sẽ ghép các vector nhúng thực thể và vector nhúng quan hệ theo cấu trúc như sau :

$$\overrightarrow{t_{ijk}} = \mathbf{W}_1[\overrightarrow{e_i}||\overrightarrow{e_j}||\overrightarrow{r_k}] \quad (4.15)$$

Với $\overrightarrow{t_{ijk}}$ là vector nhúng đại diện cho bộ ba $t_{ij}^k = (e_i, r_k, e_j)$ với e_j , và r_k lần lượt là các thực thể hàng xóm và quan hệ nối giữa đỉnh gốc e_i với đỉnh e_j , $\mathbf{W} \in \mathbb{R}^{D_k \times (2D_{\text{in}} + P_{\text{in}})}$ là ma trận trọng số thể hiện quá trình biến đổi tuyến tính từ các vector đã ghép lại với nhau thành một vector với số chiều D_k mới. Các ma trận trọng số trên được khởi tạo ngẫu nhiên theo phân phối chuẩn hoặc tái huấn luyện (pre-train) bằng mô hình TransE [5].

Tương tự với công thức 4.7 của mô hình GAT, ta cần tính hệ số chú ý của từng cạnh đối với từng đỉnh, sau đó áp dụng hàm *softmax* để chuẩn hóa hệ số lại theo công thức sau :

$$\begin{aligned} \alpha_{ijk} &= \text{softmax}_{jk}(\text{LeakyReLU}(\mathbf{W}_2 \overrightarrow{t_{ijk}})) \\ &= \frac{\exp(\text{LeakyReLU}(\mathbf{W}_2 \overrightarrow{t_{ijk}}))}{\sum_{n \in \mathcal{N}_i} \sum_{r \in \mathcal{R}_{in}} \exp(\text{LeakyReLU}(\mathbf{W}_2 \overrightarrow{t_{inr}}))} \end{aligned} \quad (4.16)$$

trong đó \mathcal{N}_i là tập hợp hàng xóm của đỉnh gốc e_i có độ sâu n_{hop} ; \mathcal{R}_{in} là tập hợp tất cả những quan hệ nằm trên đường đi (path) từ thực thể gốc e_i với thực thể $e_n \in \mathcal{N}_i$. Tương tự công thức 4.8, các vector nhúng $\overrightarrow{t_{ij}^k}$ sẽ được thu nhỏ hoặc mở rộng khi nhân với hệ số chú ý đã được chuẩn hóa :

$$\overrightarrow{e_i} = \sigma \left(\sum_{j \in \mathcal{N}_i} \sum_{k \in \mathcal{R}_{ij}} \alpha_{ijk} \overrightarrow{t_{ijk}} \right) \quad (4.17)$$

Tương tự như công thức 4.9 của *cơ chế mặt nạ chú ý*, ta sẽ ghép N_{head} đỉnh chú ý lại với nhau để ổn định quá trình học :

$$\overrightarrow{e_i} = \parallel_{h=1}^{N_{\text{head}}} \sigma \left(\sum_{j \in \mathcal{N}_i} \alpha_{ijk}^{(h)} \overrightarrow{t_{ijk}^{(h)}} \right) \quad (4.18)$$

Tương tự như các vector nhúng thực thể, các vector nhúng quan hệ cũng được nhân với một ma trận trọng số \mathbf{W}_R để thực hiện biến đổi tuyến tính các vector nhúng quan hệ P chiều lên vector nhúng có P' chiều :

$$\mathbf{R}' = \mathbf{R}\mathbf{W}^R; \quad \text{với: } \mathbf{W}^R \in \mathbb{R}^{P \times P'} \quad (4.19)$$

Đến đây, ta đã có hai ma trận $\mathbf{H}' \in \mathbb{R}^{N_e \times D'}$ và $\mathbf{R}' \in \mathbb{R}^{N_r \times P'}$ tương ứng là ma trận quan hệ và ma trận thực thể với số chiều mới. Mô hình sẽ đi qua lớp chú ý cuối cùng với đầu vào các vector nhúng quan hệ và thực thể mới như 4.10 . Tuy nhiên, nếu chúng ta thực hiện chú ý đa đỉnh trên lớp cuối cùng này để dự đoán, phép ghép chồng sẽ không còn *nhạy cảm* với cơ chế tự-chú ý. Vì vậy thay vì ghép chồng, mô hình sẽ tính trung bình và sau đó áp dụng hàm phi tuyến tính cuối cùng :

$$\vec{e}_i' = \sigma \left(\frac{1}{N_{\text{head}}} \sum_{h=1}^{N_{\text{head}}} \sum_{j \in \mathcal{N}_i} \sum_{k \in \mathcal{R}_{ij}} \alpha_{ijk}'^{(h)} \vec{t}_{ijk}'^{(h)} \right) \quad (4.20)$$

Với $\alpha_{ijk}'^{(h)}$ và $t_{ijk}'^{(h)}$ tương ứng là hệ số chú ý đã chuẩn hóa và vector nhúng đại diện cho một bộ ba (e_i, r_k, e_j) thuộc các lớp (h) khác nhau.

Đến đây, mô hình KBGAT đã áp dụng giống như mô hình GAT 4.3 nhưng ta bổ sung thêm thông tin vector nhúng thực thể và thông tin các đỉnh hàng xóm cách n_{hop} bậc, ta đã thu được ma trận nhúng thực thể $\mathbf{E}'' \in \mathbb{R}^{N_e \times D''}$, và ma trận nhúng quan hệ $\mathbf{R}'' \in \mathbb{R}^{N_r \times P''}$. Tuy nhiên, sau khi trải qua quá trình học ma trận nhúng mới, ma trận nhúng thực thể \mathbf{E}'' mất đi thông tin nhúng khởi tạo ban đầu do xảy ra hiện tượng **biến mất đạo hàm** (vanishing gradients). Để giải quyết vấn đề này, mô hình sử dụng kỹ thuật học còn sót (residual learning) sẽ cho nhân ma trận nhúng khởi tạo ban đầu \mathbf{E} với một ma trận trọng số $\mathbf{W}^E \in \mathbb{R}^{D_{\text{in}} \times D''}$ để tạo thành ma trận nhúng mới rồi cộng trực tiếp ma trận nhúng đó vào để đảm bảo thông tin nhúng khởi tạo trong quá trình huấn luyện :

$$\mathbf{H} = \mathbf{W}^E \mathbf{E} + \mathbf{E}'' \quad (4.21)$$

Cuối cùng, các bộ dữ liệu huấn luyện sẽ được lấy mẫu để tạo ra bộ ba hợp lệ (valid triple), và bộ ba không hợp lệ (invalid triple) tương tự như mô hình TransE được trình bày ở trên để học được các vector nhúng. Tuy nhiên công thức tính sự khác biệt giữa các vector nhúng được chuẩn hóa L1 theo công thức sau : $d_{t_{ij}} = \|\vec{h}_i + \vec{g}_k - \vec{h}_j\|_1$.

Tương tự, chúng tôi huấn luyện sử dụng hàm mất mát-lề :

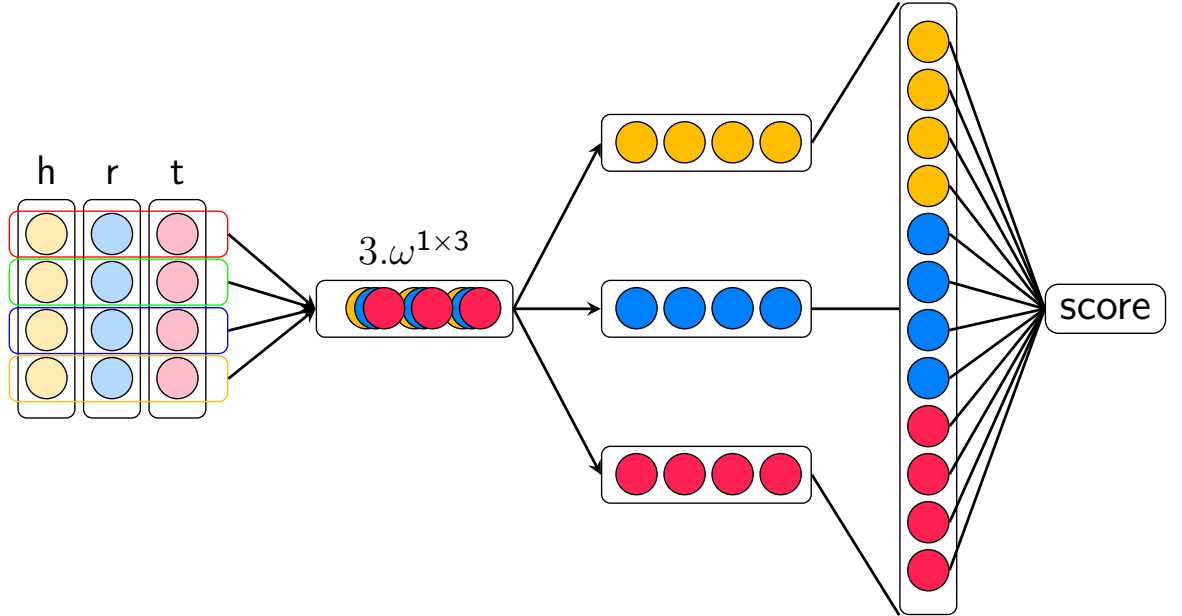
$$L(\Omega) = \sum_{t_{ij} \in S} \sum_{t'_{ij} \in S'} \max\{d_{t'_{ij}} - d_{t_{ij}} + \gamma, 0\} \quad (4.22)$$

trong đó $\gamma > 0$ là tham số lề, S là tập hợp bộ ba chuẩn (valid triple), và S' là tập hợp của ba lỗi (invalid triple) theo công thức sau :

$$S' = \underbrace{\{t'_{ij} | e'_i \in \mathcal{E} \setminus e_i\}}_{\text{thay thế thực thể đầu}} \cup \underbrace{\{t'_{ij} | e'_j \in \mathcal{E} \setminus e_j\}}_{\text{thay thế thực thể đuôi}} \quad (4.23)$$

Đầu ra của mô hình KBGAT là các vector nhúng thực thể và các vector nhúng quan hệ, sau đó các vector nhúng này tiếp tục đi qua mô hình ConvKB để tiến hành dự đoán.

4.4.3 Mô hình dự đoán ConvKB



Hình 4.11: Minh họa các lớp giải mã của mô hình ConvKB với 3 filter

Sau khi chúng ta biểu diễn các thực thể và quan hệ lên không gian số chiều thấp, mô hình sẽ sử dụng [33] làm mô hình để phân tích các đặc trưng toàn cục của một bộ ba t_{ijk} qua mỗi chiều để khái quát hóa các đặc trưng biến đổi của mô hình bằng các lớp tính tích chập. Hàm tính điểm số với những ánh xạ đặc trưng được tính theo công thức sau :

$$f(t_{ijk}) = \left(\prod_{m=1}^{\Omega} \text{ReLU}([\vec{e}_i, \vec{r}_k, \vec{h}_j] * \omega^m) \right) \cdot \mathbf{W} \quad (4.24)$$

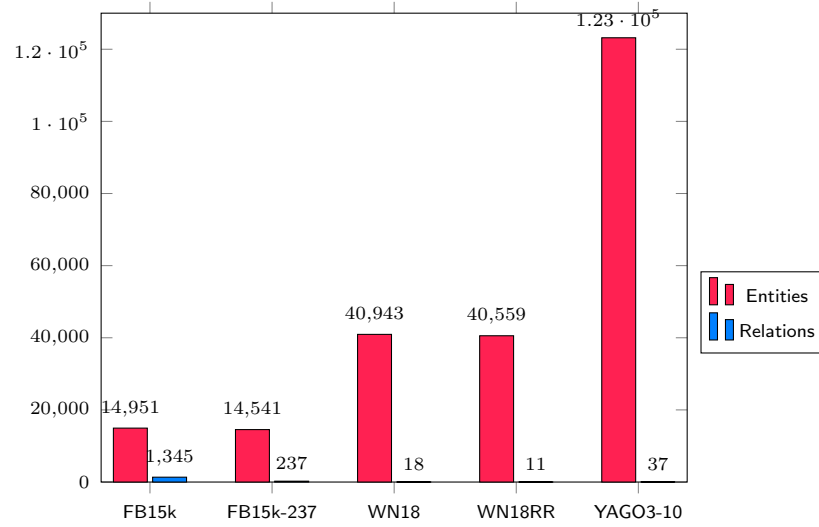
trong đó ω^m thể hiện bộ lọc tích chập thứ m - th, ω là siêu tham số về số lượng lớp tích chập, $*$ là thao tác thực hiện tính tích chập, và $\mathbf{W} \in \mathbb{R}^{\Omega k \times 1}$ biểu thị ma trận biến đổi tuyến tính được sử dụng để tính kết quả cuối cùng của bộ ba. Mô hình được huấn luyện bằng lề-mềm (soft-margin) như sau :

$$\mathcal{L} = \sum_{t_{ij}^k \in \{S \cup S'\}} \log(1 + \exp(l_{t_{ij}^k} \cdot f(t_{ij}^k))) + \frac{\lambda}{2} \|\mathbf{W}\|_2^2 \quad (4.25)$$

trong đó $l_{t_{ij}^k} = \begin{cases} 1 & \text{for } t_{ij}^k \in S \\ -1 & \text{for } t_{ij}^k \in S' \end{cases}$

Đầu ra cuối cùng của mô hình ConvKB là điểm số xếp hạng tương ứng của từng dự đoán.

Chương 5. THỰC NGHIỆM



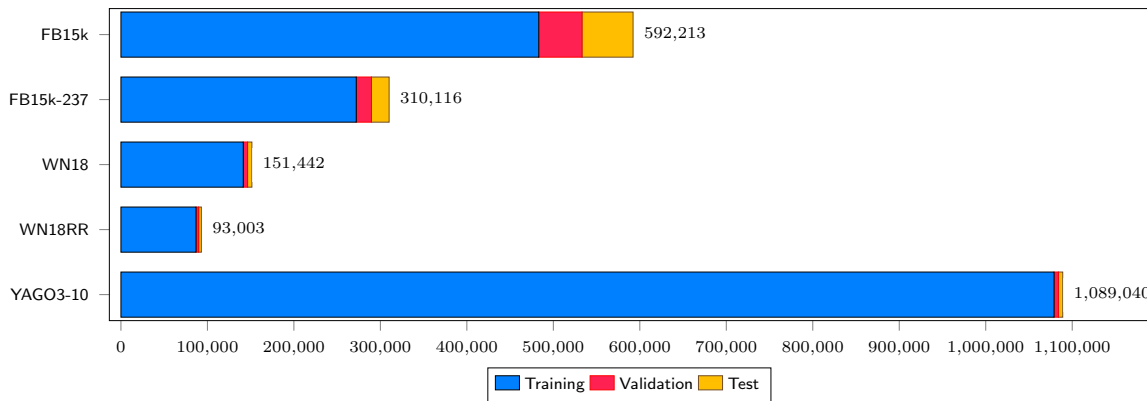
Trong phần này chúng tôi mô tả lại các bộ dữ liệu mà chúng tôi dùng để thực nghiệm đánh giá phương pháp của chúng tôi cùng với so sánh với các kết quả khác của các công trình nổi bật khác được báo cáo trong [Bảng 4.1](#). Ngoài ra chúng tôi cũng cố gắng đánh giá hai đề xuất của chúng tôi trong việc thêm một lượng tri thức mới vào đồ thị bằng cách chúng tôi xem tập test là một lượng tri thức mới cần thêm vào và dùng tập validation để đánh giá lại phương pháp của chúng tôi. Kết quả chi tiết được báo cáo ở [Bảng 5.2](#), [Bảng 5.3](#).

5.1 Các tập dữ liệu huấn luyện

Trong thí nghiệm của chúng tôi, chúng tôi thực hiện trên bốn tập dữ liệu phổ biến là FB15k, FB15-237 ([\[42\]](#)), WN18 và WN18RR ([\[12\]](#)). Mỗi bộ dữ liệu đều bao gồm 3 thành phần : tập dữ liệu huấn luyện (train), kiểm định (valid) và tập dữ liệu kiểm thử (test). Thông tin chi tiết về các tập dữ liệu được trình bày ở [bảng 5.1](#). Mỗi tập dữ liệu này là tập hợp các bộ ba $\langle head, relation, tail \rangle$. FB15k, WN18 là các bộ dữ liệu trích xuất từ bộ dữ liệu lớn gốc là FreeBase và

Dataset	Entities	Relations	# Edges		
			Training	Validation	Test
FB15k	14,951	1,345	483,142	50,000	59,071
FB15k-237	14,541	237	272,115	17,535	20,466
WN18	40,943	18	141,442	5,000	5,000
WN18RR	40,559	11	86,835	3,034	3,134
YAGO3-10	123,182	37	1,079,040	5,000	5,000

Bảng 5.1: Thông tin các tập dữ liệu



WordNet bao gồm rất nhiều quan hệ đảo ngược, nên nó có thể dễ dàng có thể đoán được với hầu hết các bộ ba. Bằng cách loại bỏ các quan hệ này, hai bộ dữ liệu FB15k-237 và WN18RR được tạo ra để giúp bộ dữ liệu thể hiện đúng dữ liệu trong thực tế để nghiên cứu về nhiệm vụ dự đoán liên kết.

5.1.1 Bộ dữ liệu FB15k

Bộ dữ liệu này được tạo bởi nhóm nghiên cứu A. Bordes, N. Usunier [5] bằng cách trích xuất từ bộ dữ liệu Wikilinks database¹. Wikilinks database thu thập các siêu liên kết (hyperlinks) đến Wikipedia gồm 40 triệu lượt đề cập trên 3 triệu thực thể, họ trích xuất tất cả các dữ kiện liên quan đến một thực thể nhất định có hơn 100 lần được đề cập đến bởi các tài liệu khác cùng với tất cả các dữ kiện liên quan đến thực thể đó (bao gồm cả những thực thể con được nhắc đến trong tài liệu Wikipedia đó), ngoại trừ những thông tin như: ngày tháng, danh

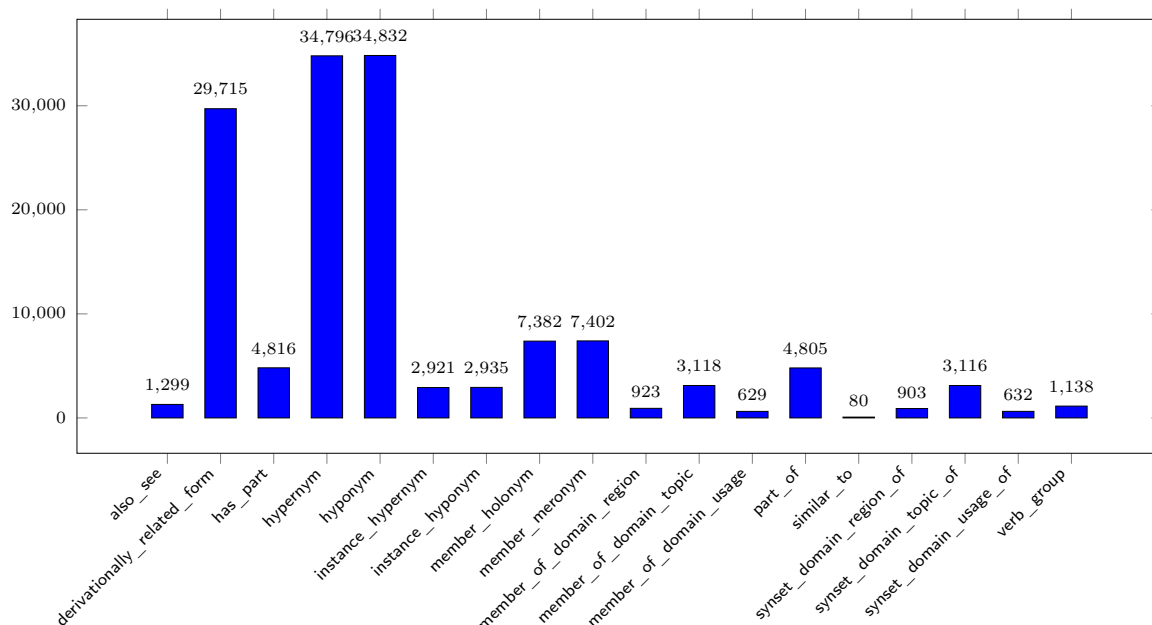
¹<https://code.google.com/archive/p/wiki-links/>

từ riêng, v.v ... Họ cũng chuyển đổi các đỉnh có bậc n được biểu diễn thành các nhóm các cạnh nhị phân tức là liệt kê các cạnh và quan hệ của mọi đỉnh.

5.1.2 Bộ dữ liệu FB15k-237

Bộ dữ liệu này là một tập hợp con của FB15k được xây dựng bởi Toutanova và Chen [42] lấy cảm hứng từ quan sát rằng FB15k bao gồm dữ liệu thử nghiệm được các mô hình nhìn thấy tại thời điểm đào tạo (test leakage). Trong FB15k, vấn đề này là do sự hiện diện của các quan hệ gần giống nhau hoặc nghịch đảo của nhau. FB15k-237 được xây dựng để trở thành một tập dữ liệu thách thức hơn: các tác giả đã chọn các dữ kiện liên quan đến 401 quan hệ xuất hiện nhiều nhất và loại bỏ tất cả các quan hệ tương đương hoặc nghịch đảo. Họ cũng đảm bảo rằng không có thực thể nào được kết nối trong tập huấn luyện cũng được liên kết trực tiếp trong tập test và validation.

5.1.3 Bộ dữ liệu WN18



Bộ dữ liệu này được giới thiệu bởi các tác giả của TransE [5], được trích xuất từ WordNet², một bản thể học ngôn ngữ KG có nghĩa là cung cấp một từ điển/từ đồng nghĩa để hỗ trợ NLP và phân tích văn bản tự động. Trong

²<https://wordnet.princeton.edu/>

WordNet, các thực thể tương ứng với các tập hợp (*word senses*) và các quan hệ đại diện cho các kết nối từ vựng của chúng (ví dụ: “hypernym”). Để xây dựng WN18, các tác giả đã sử dụng WordNet làm điểm bắt đầu và sau đó lặp đi lặp lại lọc ra các thực thể và mối quan hệ với quá ít lần được đề cập.

5.1.4 Bộ dữ liệu WN18RR

Bộ dữ liệu này là một tập hợp con của WN18 được xây dựng bởi DeŠmers et al.[11], cũng là người giải quyết vấn đề rò rỉ thử nghiệm (test leakage) trong WN18. Để giải quyết vấn đề đó, họ xây dựng tập dữ liệu WN18RR thách thức hơn nhiều bằng cách áp dụng một phương pháp tương tự được sử dụng cho FB15k-237 [42].

5.2 Các độ đo

Trong phần này chúng tôi mô tả lại các phương pháp đánh giá (độ đo), môi trường thực hiện cũng như các tập dữ liệu mà chúng tôi sử dụng để đánh giá phương pháp của mình. Các phương pháp đánh giá (độ đo) này cũng phổ biến nó được đánh giá cho hầu hết các mô hình dự đoán liên kết trên đồ thị. Chúng tôi tiến hành so sánh với bốn phương pháp nổi bật khác được báo cáo trong [37].

5.2.0.1 Độ đo Hit@K (H@K)

Đó là tỷ lệ các dự đoán đúng mà rank nhỏ hơn hoặc bằng ngưỡng K :

$$H@K = \frac{|\{q \in Q : rank(q) \leq K\}|}{|Q|}$$

5.2.0.2 Mean Rank (MR)

Đây là giá trị trung bình của rank thu được cho một dự đoán chính xác. Càng nhỏ thì mô hình càng chính xác:

$$MR = \frac{1}{|Q|} \sum_{q \in Q} rank(q)$$

Trong đó $|Q|$ là độ lớn của tập hợp các câu hỏi bằng độ lớn của tập test hoặc validation. Khi dự đoán chúng tôi dự đoán cả head và tail cho một dòng tương

ứng trong tập dữ liệu thử nghiệm. Ví dụ chúng tôi sẽ dự đoán $\langle ?, relation, tail \rangle$ và $\langle head, relation, ? \rangle$ cho 1 dòng tương ứng, q thể hiện cho câu hỏi chúng tôi dự đoán và $rank(q)$ thể hiện cho kết quả đúng của câu hỏi đứng ở vị trí thứ mấy trong xếp hạng của chúng tôi sau đó lấy trung bình rank của các dự đoán head và tail. Rõ ràng độ đo này nằm giữa $[1, | \text{số lượng các entity} |]$ do có tối đa n cạnh nối 1 đỉnh tới $n - 1$ đỉnh còn lại và thêm cạnh nối tới chính đỉnh nó (cạnh khuyên). Và độ đo này dễ bị ảnh hưởng bởi nhiễu vì có những quan hệ có những thực thể được xếp hạng gần cuối. Để giải quyết vấn đề này nhóm chúng tôi và các nhóm nghiên cứu khác sử dụng thêm độ đo Mean Reciprocal Rank (MMR)

5.2.0.3 Mean Reciprocal Rank (MMR)

Đây là xếp hạng đối ứng trung bình, là nghịch đảo của giá trị trung bình của rank thu được cho một dự đoán chính xác ở trên. Và càng lớn thì mô hình càng chính xác. Do độ đo này lấy nghịch đảo của các rank nên tránh được vấn đề nhiễu của độ đo MR ở trên.

$$MRR = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{rank(q)}$$

5.3 Phương pháp huấn luyện

5.3.1 Huấn luyện trên mô hình KBGAT

Đầu tiên chúng tôi khởi tạo các vector nhúng bằng mô hình TransE ([5]). Để tạo ra các bộ ba không hợp lệ, chúng tôi thay thế các thực thể đầu và thực thể đuôi bằng một thực thể khác được lấy ngẫu nhiên trong tập thực thể.

Sau đó chúng tôi chia ra làm hai phần huấn luyện, phần đầu tiên được xem như mã hóa (encoder) giúp biến đổi các vector nhúng khởi tạo ban đầu thành các vector nhúng mới tổng hợp thông tin các nút lân cận bằng mô hình KBGAT để tạo ra các vector nhúng của thực thể và quan hệ. Phần thứ hai được xem như quá trình giải mã (decoder) để thực hiện nhiệm vụ dự đoán, bằng cách lấy thêm thông tin của n-hop giúp chúng tôi tổng hợp thêm thông tin từ các thực thể lân cận, ngoài ra chúng tôi còn sử dụng quan hệ phụ trợ để tổng hợp thêm thông tin hàng xóm trong đồ thị thưa. Chúng tôi sử dụng hàm tối ưu Adam với tốc độ học $\mu = 0.001$. Số chiều cuối cùng của cả thực thể và quan hệ đều bằng

200. Cụ thể các siêu tham số ưu được tìm kiếm bằng thuật toán tìm kiếm lưới (grid search) được trình bày ở [Phụ lục A](#).

5.4 Kết quả thực nghiệm

Như đã nói trước đây với mô hình dựa trên luật của chúng tôi hoàn toàn có thể thực hiện trên một laptop với cấu hình thông thường. Trong thí nghiệm của chúng tôi cấu hình máy để thực thi như sau: T480, core i5 8th Gen, ram 16GB, 4 core 8 thread. Mã nguồn thực thi được viết bằng ngôn ngữ Python phiên bản 3.6 và dùng các hàm hỗ trợ có sẵn trong Python với không một thư viện bên thứ ba nào. Thí nghiệm được thực hiện với bốn tập dữ liệu phổ biến là FB15k, FB15-237, WN18 và WN18RR. Thông tin chi tiết các bộ dữ liệu này được mô tả ở [Mục 5.1](#) các tập dữ liệu huấn luyện.

Như mô tả ở phần [Thuật toán 2](#), thuật toán AnyBURL này sẽ học các luật được sinh ra trong một khoảng thời gian nhất định do người dùng cấu hình. Ở đây chúng tôi chọn cấu hình thời gian là 1000 giây tương đương khoảng 17 phút đào tạo, với độ bão hòa (SAT) 0.85, độ tin cậy Q 0.05, kích thước mẫu S ($\frac{1}{10}$ tập huấn luyện). Với cấu hình như vậy mô hình phiên bản Python của chúng tôi cho kết quả tương đương với phiên bản Java nhóm tác giả Meilicke, Christian et al. [27] với cấu hình tương tự nhưng thời gian training là 100 giây. Sự khác biệt về thời gian học tập ở đây chủ yếu là do hiệu năng của hai ngôn ngữ Python và Java. Ở đây chúng tôi chọn ngôn ngữ Python vì nó được dùng làm ngôn ngữ chính cho nhiều mô hình trí tuệ nhân tạo gần đây, và cũng thuận tiện cho chúng tôi khi so sánh hiệu năng cũng như đánh giá với các phương pháp học sâu khác đa số được viết bằng Python.

	FB15k				FB15k-237			
	H@1	H@10	MR	MRR	H@1	H@10	MR	MRR
ComplEx	81.56	90.53	34	0.848	25.72	52.97	202	0.349
TuckER	72.89	88.88	39	0.788	25.90	53.61	162	0.352
TransE	49.36	84.73	45	0.628	21.72	49.65	209	0.31
RoteE	73.93	88.10	42	0.791	23.83	53.06	178	0.336
ConvKB	59.46	84.94	51	0.688	21.90	47.62	281	0.305
KBGAT	70.08	91.64	38	0.784	36.06	58.32	211	0.4353
AnyBURL	79.13	82.30	285	<u>0.824</u>	20.85	42.40	490	0.311

Bảng 5.2: Kết quả thực nghiệm trên tập FB15k, FB15k-237

	WN18				WN18RR			
	H@1	H@10	MR	MRR	H@1	H@10	MR	MRR
ComplEx	94.53	95.50	3623	0.349	42.55	52.12	4909	0.458
TuckER	94.64	95.80	510	0.951	42.95	51.40	6239	0.459
TransE	40.56	94.87	279	0.646	2.79	94.87	279	0.646
RoteE	94.30	96.02	274	0.949	42.60	57.35	3318	0.475
ConvKB	93.89	95.68	413	0.945	38.99	50.75	4944	0.427
KBGAT					35.12	57.01	<u>1974</u>	0.4301
AnyBURL	93.96	95.07	230	0.955	44.22	54.40	2533	<u>0.497</u>

Bảng 5.3: Kết quả thực nghiệm trên tập WN18, WN18RR

Bảng 5.2, và Bảng 5.3 mô tả các kết quả thực nghiệm của chúng tôi với các độ đo $H@K$ cùng với các kết quả thực nghiệm của các phương pháp khác được đề cập trong khảo sát [37]

		AnyBURL	Batch edge AnyBURL	Edge AnyBURL
FB-15k	hit@10	82.22	82.48	83.08
	MR	285	250	220
	MRR	0.824	0.853	0.866
FB15k-237	hit@10	42.40	43.40	43.51
	MR	490	472	441
	MRR	0.311	0.353	0.377
WN18	hit@10	95.07	95.09	95.19
	MR	230	229	228
	MRR	0.955	0.955	0.956
WN18RR	hit@10	54.40	54.63	54.70
	MR	2533	2346	2215
	MRR	0.497	0.553	0.581

Bảng 5.4: Kết quả độ chính xác hai chiến lược thêm tri thức mới

Bảng 5.4 mô tả các kết quả thực nghiệm của chúng tôi với hai chiến lược thêm tri thức mới vào đồ thị. Chúng tôi đánh giá trên tổng số luật được sinh ra, và số luật có độ tin cậy $\geq 50\%$ và $\geq 80\%$.

		Batch edge AnyBURL	Edge AnyBURL
FB15k	num rule	1011	1367
	confidence 50%	416 (41,14%)	1185 (86,69%)
	confidence 80%	284 (28, 09%)	481 (35,18%)
FB15k-237	num rule	1120	756
	confidence 50%	244 (21,79%)	660 (87,30%)
	confidence 80%	95 (8,48%)	162 (21,43%)
WN18	num rule	533	260
	confidence 50%	270 (38, 46 %)	252 (96,92%)
	confidence 80%	240 (34,19%)	225 (86,54%)
WN18RR	num rule	439	106
	confidence 50%	110 (25,05%)	102 (96,22%)
	confidence 80%	83 (18,91%)	85 (81,19%)

Bảng 5.5: Kết quả đánh giá về số luật hai chiến lược thêm tri thức mới

Chương 6. KẾT LUẬN

Trong phần này, chúng tôi đã trình bày các kết quả đạt được của mô hình đề xuất, kèm theo các phân tích chi tiết trên từng tập dữ liệu khác nhau nhằm làm rõ những điểm mạnh cũng như các hạn chế còn tồn tại. Qua đó, chúng tôi xác định các hướng nghiên cứu tiềm năng để cải thiện mô hình trong tương lai.

Mặc dù phương pháp dựa trên luật của chúng tôi cho thấy hiệu suất so sánh được với các mô hình học sâu hiện đại (state-of-the-art), đồng thời có ưu thế rõ rệt về thời gian huấn luyện - chỉ khoảng 17 phút so với hàng giờ ở các mô hình học sâu - điều này không có nghĩa rằng các mô hình học sâu không đáng được nghiên cứu. Trái lại, qua phân tích hiệu quả trên các tập dữ liệu khác nhau, chúng tôi nhận thấy rằng với những bộ dữ liệu có đa dạng quan hệ như FreeBase, mô hình KBGAT sử dụng cơ chế chú ý cho kết quả vượt trội so với trên các tập như WordNet, vốn có số lượng quan hệ ít hơn. Điều này cho thấy tiềm năng của việc khai thác các cơ chế học sâu phù hợp với từng đặc điểm của dữ liệu.

Điều này chứng tỏ cơ chế chú ý bổ sung thêm thông tin vector nhúng quan hệ giúp học được các cấu trúc của đồ thị tốt hơn trên các tập dữ liệu nhiều loại quan hệ. Đối với tập dữ liệu có nhiều mẫu tương tự và nghịch đảo như FB15k và WN18RR, mô hình dựa trên luật AnyBURL đạt được kết quả vượt trội, trong khi với phương pháp học sâu mô hình chỉ đạt kết quả trung bình so với các phương pháp khác. Mô hình dựa trên luật AnyBURL có ưu thế tốt hơn trên các tập dữ liệu FB15k và WN18RR, tuy nhiên với các tập dữ liệu đã loại bỏ các thông tin tương tự hoặc nghịch đảo như FB15-237 và WN18RR thì phương pháp dựa trên luật tỏ ra kém hiệu quả hơn vì mô hình này học dựa trên các đường đi hoặc liên kết đã xảy ra. Với phương pháp học sâu, mô hình biểu diễn các quan hệ và thực thể lên không gian để học được mối quan hệ giữa chúng, chính vì vậy mà mô hình học sâu đạt kết quả tốt hơn trên các tập dữ liệu như FB15k-237 và WN18RR so với các tập dữ liệu như FB15k và WN18. Một trong

những ưu điểm của phương pháp dựa trên luật đó là các luật được sinh ra có thể lý giải được trong quá trình học, và có thời gian học vượt trội so với các phương pháp khác. Tuy nhiên, sau quá trình học phương pháp dựa trên luật phải duyệt qua tất cả các luật đã học mới có thể đưa ra được dự đoán. Đây là một điểm mà các phương pháp học sâu thể hiện tốt hơn, vì thông qua các trọng số đã học, mô hình học sâu KBGAT thông qua các lớp tính toán có thể biến đổi đầu vào thành các kết quả xác xuất dự đoán nhanh hơn. Nhược điểm với phương pháp học sâu là quá trình học không thể lý giải được, hơn nữa chi phí huấn luyện rất tốn kém. Đối với hai thuật toán mở rộng của chúng tôi trong việc thêm tri thức mới vào đồ thị chúng tôi nhận thấy rằng là vượt trội hoàn toàn so với các phương pháp học sâu.

Quá trình nhúng đồ thị giúp biểu diễn các đặc trưng của thực thể, quan hệ hay các đặc trưng của đồ thị tri thức thành các vector có số chiều thấp hơn (Mục 4.1), nhưng trong thực tế, một tri thức được biểu diễn bằng những thực thể và những quan hệ là hoàn toàn độc lập với nhau nên cần phải biểu diễn chúng thành các vector có số chiều khác nhau, tỷ lệ số chiều giữa chúng cũng là vấn đề quan trọng cần nghiên cứu. Bên cạnh đó, trong thế giới thực, yếu tố thời gian là thông tin quan trọng có thể thay đổi hoàn toàn ý nghĩa của một tri thức. Vì vậy, cách cơ chế chú ý bổ sung thêm thông tin thời gian là một trong những hướng nghiên cứu chúng tôi cần áp dụng để đảm bảo tính đúng đắn của đồ thị tri thức. Đối với phương pháp học trên luật AnyBURL, gần đây nhánh học tăng cường (reinforcement learning) khá phát triển và nhóm tác giả Meilicke, Christian and Chekol [26] gần đây cũng đã có 1 nghiên cứu để tối ưu hóa lại phương pháp AnyBURL này. Chúng tôi cũng có ý định nghiên cứu về hướng này và cố gắng báo cáo lại trong một tương lai không xa.

TÀI LIỆU TRÍCH DẪN

- [1] Franz Baader and Tobias Nipkow. *Term rewriting and all that*. Cambridge university press, 1999.
- [2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.
- [3] Shumeet Baluja, Rohan Seth, Dharshi Sivakumar, Yushi Jing, Jay Yagnik, Shankar Kumar, Deepak Ravichandran, and Mohamed Aly. Video suggestion and discovery for youtube: taking random walks through the view graph. In *Proceedings of the 17th international conference on World Wide Web*, pages 895–904, 2008.
- [4] Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. Netgan: Generating graphs via random walks. *arXiv preprint arXiv:1803.00816*, 2018.
- [5] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795, 2013.
- [6] Samuel R Buss. An introduction to proof theory. *Handbook of proof theory*, 137:1–78, 1998.
- [7] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1616–1637, 2018.

- [8] Shaosheng Cao, Wei Lu, and Qionghai Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international conference on information and knowledge management*, pages 891–900, 2015.
- [9] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers. *arXiv preprint arXiv:1911.03584*, 2019.
- [10] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. Multi-head attention: Collaborate instead of concatenate. *arXiv preprint arXiv:2006.16362*, 2020.
- [11] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. *arXiv preprint arXiv:1707.01476*, 2017.
- [12] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [13] Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. Amie: association rule mining under incomplete evidence in ontological knowledge bases. In *WWW '13*, 2013.
- [14] Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M Suchanek. Fast rule mining in ontological knowledge bases with amie. *The VLDB Journal*, 24(6):707–730, 2015.
- [15] Google. *Introducing the Knowledge Graph: things, not strings*, 2020 (truy cập ngày 27/08/2020).
- [16] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018.
- [17] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

- [18] Lingbing Guo, Zequn Sun, and Wei Hu. Learning to exploit long-term relational dependencies in knowledge graphs. *arXiv preprint arXiv:1905.04914*, 2019.
- [19] Wilfrid Hodges et al. *A shorter model theory*. Cambridge university press, 1997.
- [20] John J Hopfield. Hopfield network. *Scholarpedia*, 2(5):1977, 2007.
- [21] Sergey Ivanov and Evgeny Burnaev. Anonymous walk embeddings. *arXiv preprint arXiv:1805.11921*, 2018.
- [22] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S Yu. A survey on knowledge graphs: Representation, acquisition and applications. *arXiv preprint arXiv:2002.00388*, 2020.
- [23] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.
- [24] Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, pages 319–345. Springer, 1999.
- [25] Johann A. Makowsky. Why horn formulas matter in computer science: Initial structures and generic examples. *Journal of Computer and System Sciences*, 34(2-3):266–292, 1987.
- [26] Christian Meilicke, Melisachew Wudage Chekol, Manuel Fink, and Heiner Stuckenschmidt. Reinforced anytime bottom up rule learning for knowledge graph completion. *arXiv preprint arXiv:2004.04412*, 2020.
- [27] Christian Meilicke, Melisachew Wudage Chekol, Daniel Ruffinelli, and Heiner Stuckenschmidt. Anytime Bottom-Up Rule Learning for Knowledge Graph Completion, 2019.
- [28] Christian Meilicke, Melisachew Wudage Chekol, Daniel Ruffinelli, and Heiner Stuckenschmidt. Anytime bottom-up rule learning for knowledge graph completion. In *IJCAI*, pages 3137–3143, 2019.

- [29] Christian Meilicke, Manuel Fink, Yanjie Wang, Daniel Ruffinelli, Rainer Gemulla, and Heiner Stuckenschmidt. Fine-grained evaluation of rule-and embedding-based systems for knowledge graph completion. In *International Semantic Web Conference*, pages 3–20. Springer, 2018.
- [30] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
- [31] Seyedeh Fatemeh Mousavi, Mehran Safayani, Abdolreza Mirzaei, and Hoda Bahonar. Hierarchical graph embedding in vector space by graph pyramid. *Pattern Recognition*, 61:245–254, 2017.
- [32] Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. Learning attention-based embeddings for relation prediction in knowledge graphs. *arXiv preprint arXiv:1906.01195*, 2019.
- [33] Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. A novel embedding model for knowledge base completion based on convolutional neural network. *arXiv preprint arXiv:1712.02121*, 2017.
- [34] Stefano Ortona, Venkata Vamsikrishna Meduri, and Paolo Papotti. Robust discovery of positive and negative rules in knowledge bases. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 1168–1179. IEEE, 2018.
- [35] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.
- [36] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models. *arXiv preprint arXiv:1906.05909*, 2019.
- [37] Andrea Rossi, Donatella Firmani, Antonio Matinata, Paolo Merialdo, and Denilson Barbosa. Knowledge graph embedding for link prediction: A comparative analysis. *arXiv preprint arXiv:2002.00819*, 2020.

- [38] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *Advances in neural information processing systems*, pages 3856–3866, 2017.
- [39] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using gpu model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.
- [40] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077, 2015.
- [41] Andrew Tao, Karan Sapra, and Bryan Catanzaro. Hierarchical multi-scale attention for semantic segmentation. *arXiv preprint arXiv:2005.10821*, 2020.
- [42] Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66, 2015.
- [43] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. International Conference on Machine Learning (ICML), 2016.
- [44] Johan Ugander, Brian Karrer, Lars Backstrom, and Cameron Marlow. The anatomy of the facebook social graph. *arXiv preprint arXiv:1111.4503*, 2011.
- [45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.
- [46] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero,

- Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.
- [47] Bộ Y Tế Việt-Nam. *Trí nhớ và chú ý*, 2020 (truy cập ngày 26/08/2020).
- [48] Thanh Vu, Tu Dinh Nguyen, Dat Quoc Nguyen, Dinh Phung, et al. A capsule network-based embedding model for knowledge graph completion and search personalization. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2180–2189, 2019.
- [49] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225–1234, 2016.
- [50] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763, 2019.

Phụ lục A. CÁC SIÊU THAM SỐ TỐI ƯU

Trong phần này, chúng tôi trình bày tập hợp các siêu tham số tối ưu được sử dụng cho cả hai mô hình: mô hình dựa trên cơ chế chú ý và mô hình ConvKB. Quá trình tối ưu siêu tham số được thực hiện thông qua phương pháp tìm kiếm lưới (grid search) dựa trên chỉ số đánh giá Hits@10. Đối với mô hình chú ý, chúng tôi huấn luyện trên toàn bộ tập dữ liệu mà không thực hiện chia nhỏ. Trong khi đó, đối với mô hình ConvKB, chúng tôi áp dụng cùng một kích thước siêu tham số cho tất cả các tập dữ liệu. Chi tiết các siêu tham số được thể hiện trong bảng sau:

	μ	Weight decay	Epochs	negative ratio	Dropouts	$\alpha_{\text{LeakyReLU}}$	N_{head}	D_{final}	γ
FB15k	1e-3	1e-5	3000	2	0.3	0.2	2	200	1
FB15k-237	1e-3	1e-5	3000	2	0.3	0.2	2	200	1
WN18	1e-3	5e-6	3600	2	0.3	0.2	2	200	5
WN18RR	1e-3	5e-6	3600	2	0.3	0.2	2	200	5