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

Hoàng Minh Thanh - Phan Minh Tâm

DỰ ĐOÁN LIÊN KẾT TRONG ĐỒ THỊ PHỨC

(Knowledge Graph embedding for Linking Prediction)

KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN
CHƯƠNG TRÌNH HOÀN CHỈNH ĐẠI HỌC

Tp. Hồ Chí Minh, tháng 09/2020

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

Hoàng Minh Thanh - 18424062

Phan Minh Tâm - 18424059

DỰ ĐOÁN LIÊN KẾT TRONG ĐỒ THỊ PHỨC

(Knowledge Graph embedding for Linking Prediction)

KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN
CHƯƠNG TRÌNH HOÀN CHỈNH ĐẠI HỌC

GIÁO VIÊN HƯỚNG DẪN

Th.S Lê Ngọc Thành

Tp. Hồ Chí Minh, tháng 09/2020

Lời cảm ơn

Tôi xin chân thành cảm ơn t
sdfsdf

Mục lục

Danh sách hình

Danh sách bảng

Tóm tắt

Dùng sfdefaultdf s df s defcounter

Chương 1

Giới thiệu

Ngôn [5] ngữ để viết và trình bày báo cáo khóa luận tốt nghiệp, đồ án tốt nghiệp, thực tập tốt nghiệp (sau đây gọi chung là báo cáo) là tiếng Việt hoặc tiếng Anh. Trường hợp chọn ngôn ngữ tiếng Anh để viết và trình bày báo cáo, sinh viên cần có đơn đề nghị, được cán bộ hướng dẫn (CBHD) đồng ý và nộp cho bộ phận Giáo vụ của Khoa vào thời điểm đăng ký đề tài để xin ý kiến. Báo cáo viết và trình bày bằng tiếng Anh phải có bản tóm tắt viết bằng tiếng Việt.

Chương 2

Tổng quan

sdfsdf

2.1 Cơ sở lý thuyết

sdfsdfsdfsdf

2.2 Các nghiên cứu liên quan

Sau thời kỳ ngủ đông của AI, các kỹ thuật học sâu có một bước tiến đang kể với các mô hình CNN [21], RNN [18], Attention [25], Transformer [27] đã đạt được những kết quả rất lớn với độ chính xác cao cả trong nghiên cứu cũng như khi áp dụng vào công nghiệp. Các mô hình học sâu, cụ thể là CNN đã rất thành công để giải quyết các vấn đề về phân loại hình ảnh [17] , phân đoạn ngữ nghĩa [19] và dịch máy , chúng tạo ra một bản đồ đặc trưng biểu diễn cấu trúc thông tin dưới dạng lưới. CNN cũng đã được áp dụng vào trong cấu trúc đồ thị với mô hình GCN [20].

Chương 3

Phương pháp đề xuất

sdfsdf

3.1 AnyBURL

Anytime Bottom-Up Rule Learning for Knowledge Graph Completion [6] sdfsdf

Dùng lệnh để trích dẫn một hoặc nhiều tài liệu Lưu ý khi trích dẫn tài liệu tham khảo, cần viết câu sao cho bỏ phần trong cặp ngoặc vuông đi thì câu vẫn đầy đủ ý nghĩa. Ví dụ, thay Một ví dụ khác, thay vì viết "... như trong công trình nghiên viết "... nh'

Để chèn mã nguồn, cần dùng package listings

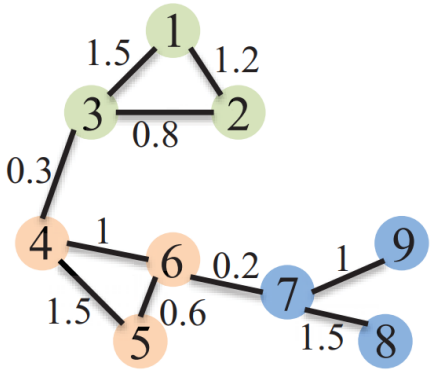
3.2 Phương pháp dựa trên học sâu - CGAT

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. Cùng với đó chúng ta sẽ trình bày phương pháp Mạng Đồ Thị Chú Ý (Graph Attention Network - GAT [26]) và phương pháp Mạng Đồ Thị Chú Ý Cơ Sở (KBAT [22]).

3.2.1 Đồ thị tri thức

Ngày nay đồ thị đã được ứng dụng vào mọi mặt của đời sống, với đồ thị về mạng xã hội (Facebook [0]) 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 [0]), 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 [0]) được Google giới thiệu năm 2012 [0], 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 biểu diễn trên đồ thị. Khai thác đồ thị hay 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.

Để tìm hiểu về đồ thị tri thức ta cần hiểu các đinh nghĩa về cơ bản đã được nhóm tác giả [1] và [3] kháo sát và phân loại như sau :



Hình 3.1: Ví dụ về đồ thị đầu vào

- **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 $f_v : V \to T^v$ và hàm ánh xạ loại cạnh: $f_e : E \to T^e$.*

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* $\mid T^v \mid = \mid T^e \mid = 1$. *Tất cả các đỉnh trong* $\mathcal{G}$ *thuộc về một loại duy nhất và tất cả các cạnh thuộc về một loại duy nhất.*

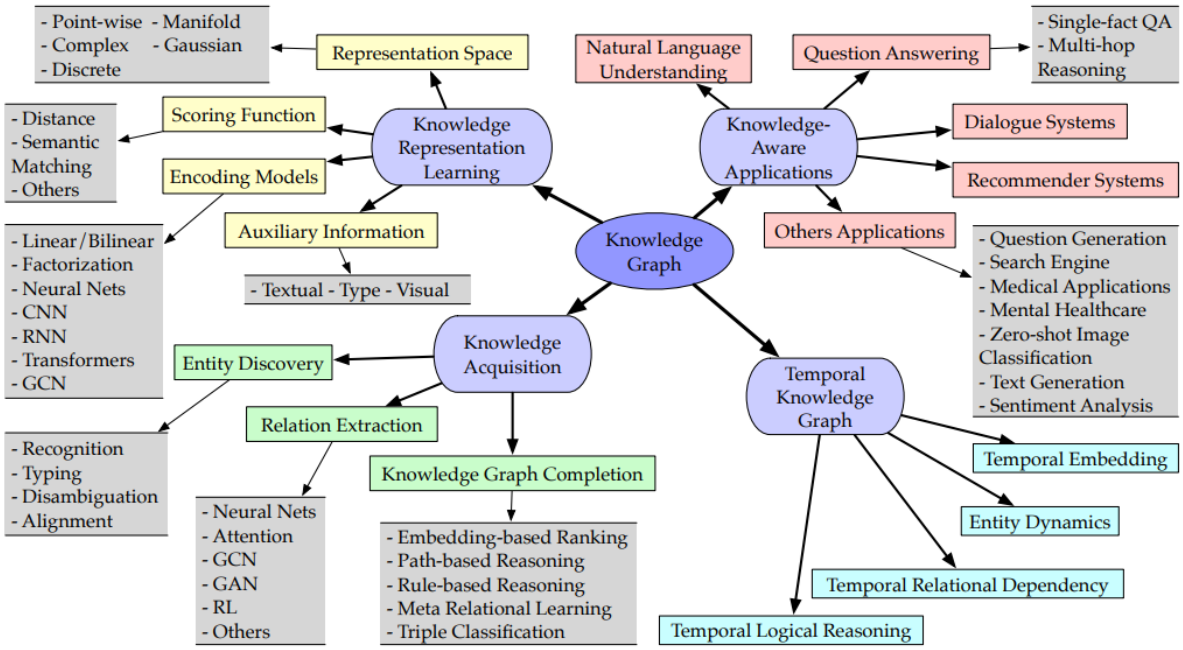
- **Định nghĩa 3 (Đồ thị không đồng nhất)** *Đồ thị không đồng nhất (heterogeneous graph) :* $\mathcal{G}_{hete} = (V, E)$ *là một đồ thị trong đó* $\mid T^v \mid > 1$ *hoặc* $\mid T^e \mid > 1$. *Tức là có nhiều hơn một loại đỉnh hoặc nhiều hơn một loại cạnh.*

- **Định nghĩa 4 (Đồ thị tri thức)** *Đồ thị tri thức (knowledge graph)* $\mathcal{G}_{know} = (V, E)$ *là một đồ thị có hướng, có các đỉnh là các thực thể (entities) và các cạnh (edges) là các sự kiện gồm bộ ba subject-property-object. Mỗi cạnh là một mẫu gồm (head entity, relation, tail entity) (ký hiệu là* $\langle h, r, t \rangle$*) biểu thị mối quan hệ của* r *từ thực thể* h *đến thực thể* t *.*

  $h, t \in V$ là các thực thể và $r \in E$ là quan hệ. Chúng ta gọi $\langle h, r, t \rangle$ một bộ ba đồ thị tri thức. Ví dụ: trong Hình 3.9 có hai bộ ba: ⟨Tom Cruise, born_in , New York⟩ và ⟨New York, state_of, U.S⟩. 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.

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 [0] 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 3.2.



Hình 3.2: Danh mục các lĩnh vực nghiên cứu trên đồ thị tri thức

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 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 phát 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 .

3.2.2 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 Donal Trump thành một vector $\overrightarrow{e_{\text{Trump}}} = [1.9_{\text{heigh}}, 0_{\text{area}}, 1_{\text{wife is Melania}}, 0_{\text{wife is Taylor}}]$. Với các đặc trưng không thể đo hoặc không có giá trị ($\cdot_{\text{area}}$) sẽ bằng 0, với các đặc trưng là giá trị mà không có độ lớn ($\cdot_{\text{wife}}$) thì ta chia thành độ lớn là xác xuất 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 các 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 $s_i^{(1)} = \left[s_{i1}^{(1)}, s_{i2}^{(1)}, \ldots, s_{i|V|}^{(1)}\right]$ biểu thị lân cận bậc nhất giữa v_i và các đỉnh khác. Lấy biểu đồ trong hình 3.1 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, $s_1^{(1)} = \Big[0, 1.2, 1.5, 0, 0, 0, 0, 0, 0\Big]$.

- **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 3.1 làm ví dụ: $s_{12}^{(2)}$ là điểm tương đồng giữa $s_1^{(1)}$ và $s_2^{(1)}$. Như đã giới thiệu trước, $s_1^{(1)} = \Big[0, 1.2, 1.5, 0, 0, 0, 0, 0, 0\Big]$ và $s_2^{(1)} =$

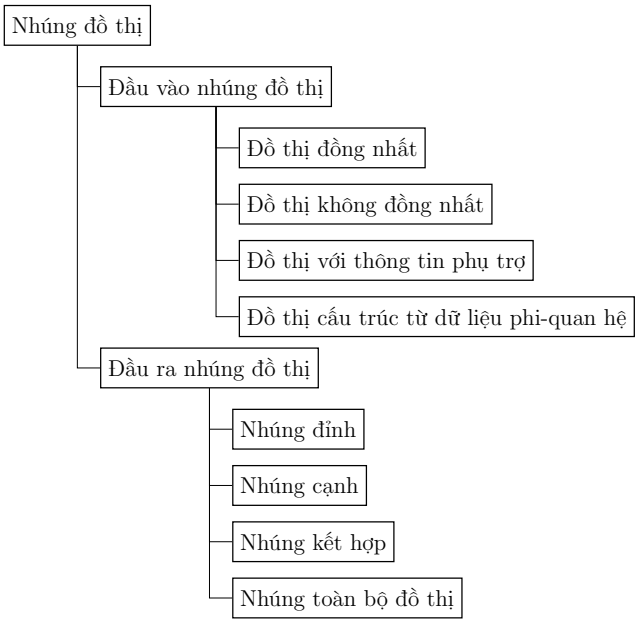
$\Big[1.2, 0, 0.8, 0, 0, 0, 0, 0, 0\Big]$. Chúng ta hãy xem xét các điểm tương đồng cosine $s_{12}^{(2)} = cosine(s_1^{(1)}, s_2^{(1)}) = 0.43$ và $s_{15}^{(2)} = cosine(s_1^{(1)}, s_5^{(1)}) = 0$. Chúng ta có thể thấy rằng lân cận bậc hai giữa v_1 và v_5 bằng 0 vì v_1 và v_5 không chia sẻ bất kỳ hàng 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 do lân cận bậc cao hơn (higher-order proximity) có thể được định nghĩa tương tự. Ví dụ, lân cận cách thứ $k-th$ giữa đỉnh v_i và v_j là sự tương đồng giữa $s_i^{(k1)}$ và $s_j^{(k1)}$. Lưu ý rằng đôi khi các giá trị gần đúng bậc cao hơn cũng được xác định bằng cách sử dụng một số số liệu khác, ví dụ: Katz Index, RootR PageRank, Adamic Adar, v.v.

- **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 $d (d \ll \mid V \mid)$, 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 3.3.

Dựa trên đầu vào nhúng ta phân loại thành các nhóm phương pháp đã khảo sát ở [1] như sau : Đồ thị đồng nhất (homogeneous graph) Đồ



Hình 3.3: Phương pháp thiết lập bài toán nhúng đồ thị

thị không đồng nhất (heterogeneous graph) Đồ thị với thông tin phụ trợ (graph with auxiliary information) Độ 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.

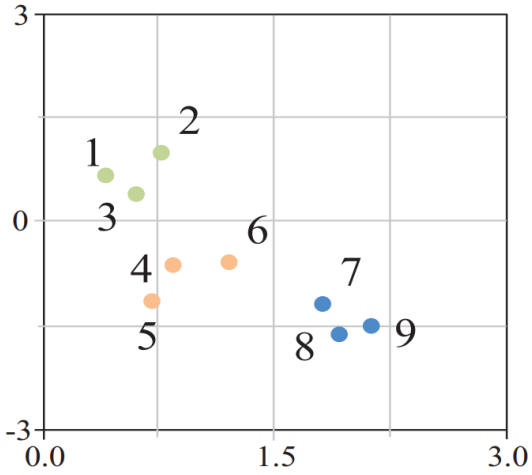
Tuy nhiên, trong một số trường hợp, các bài toán có thể liên quan đến độ chi tiết cao hơn của đồ thị, ví dụ: cặp đỉnh, đồ thị con, toàn bộ đồ

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 3.1 gồm : Nhúng Đỉnh (Node Embedding 3.4), Nhúng Cạnh (Edge Embedding 3.5), Nhúng Kết Hợp (Hybrid Embedding 3.6) và Nhúng Toàn Bộ Đồ Thị (Whole-Graph Embedding 3.7). 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.

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

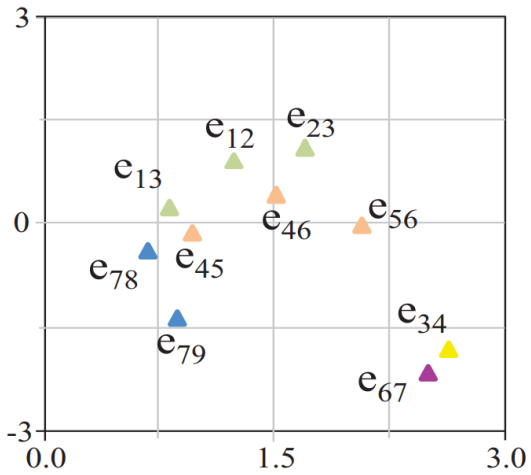


Hình 3.4: Nhúng đỉnh với từng vector thể hiện đặc trưng của từng đỉnh

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, ···) trong quá trình nhúng của chúng được đề cập trong nghiên cứu của nhóm tác giả [2].

Nhúng cạnh



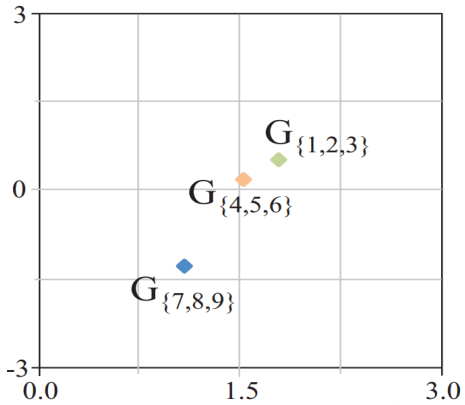
Hình 3.5: Nhúng cạnh với từng vector thể hiện đặc trưng của từng cạnh

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 đồ tri thức/dự đoán quan hệ, v.v.

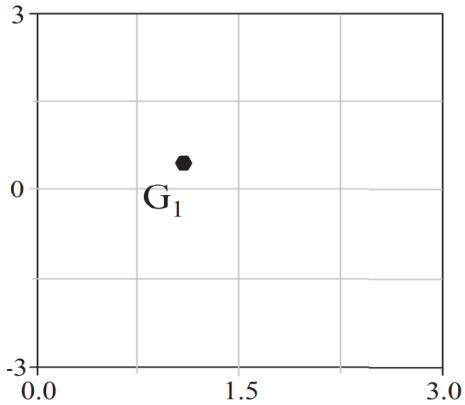
Nhúng Kết Hợp



Hình 3.6: Nhúng một cấu trúc bộ phận của đồ thị

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 3.7: Nhúng toàn bộ đồ thị

Nhúng toàn bộ đồ thị (whole-graph embedding) thường dành cho các

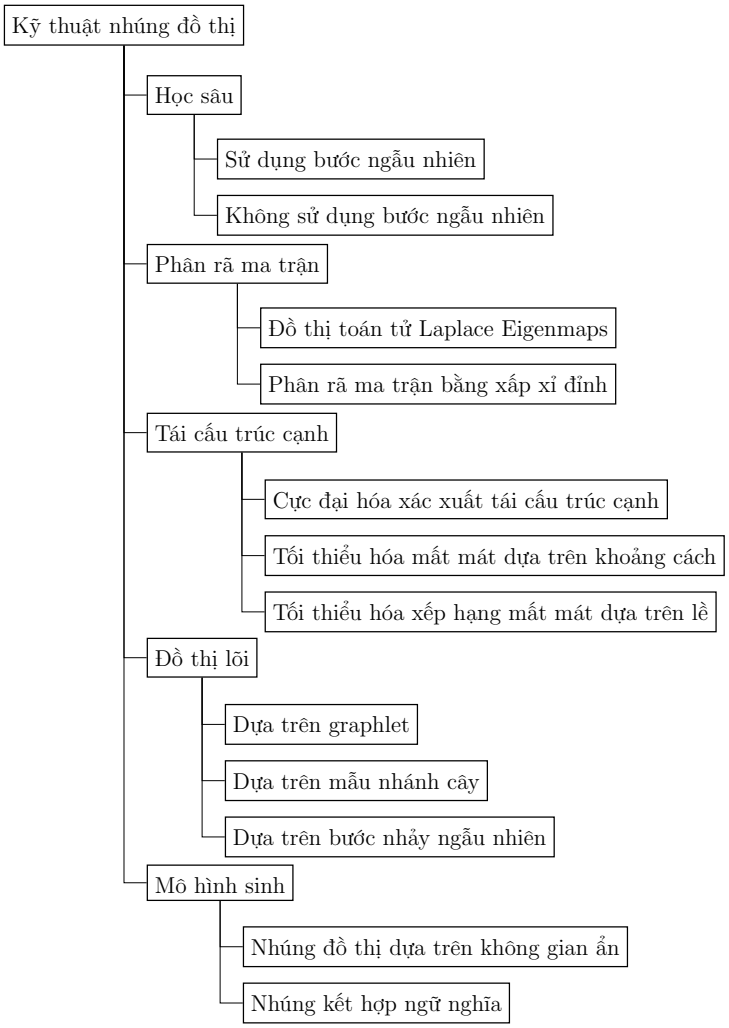
đồ thị nhỏ, chẳng hạn như protein, phân tử, v.v. Trong trường hợp này, một đồ thị được biểu diễn dưới dạng một vector và hai đồ thị tương tự được nhúng để gần nhau hơn. Việc nhúng toàn bộ đồ thị mang lại lợi ích cho nhiệm vụ phân loại đồ thị bằng cách cung cấp một giải pháp đơn giản và hiệu quả để tính 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 [4] 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.

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 đinh nghĩa các đặc tính vốn có cần được đảm bảo. 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 của nó. 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



Hình 3.8: Các kỹ thuật nhúng đồ thị

trừ đồ thị cấu trúc từ dữ liệu phi-quan hệ) và 4 loại đầu ra (Hình 3.3) 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 xuất 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 [0], LINE [0], Node2Vec [0], Anonymous Walk [0], NetGAN [0], ...

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 [5] 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 bản đồ đặc trưng, sau đó lại tiếp tục đi qua một lớp kết nối đầy đủ để tính giá trị xác xuất. Ví dụ như **ConvE** [0] : Mỗi thực thể và mối 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 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** [23], 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 $T \times 3$ bản đồ đặc trưng. Sau đó bản đồ đặc trưng lại đi qua lớp kết nối đầy đủ và trọng số $\mathbf{W}$. Kiến trúc này có thể coi là kiến trúc phân loại nhị phân.

- Mạng Hồi Qui Tuyến Tính (Recurrent Neural Networks)

  Những mô hình này sẽ cho một lớp hoặc nhiều lớp hồi 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 [0], 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 mối 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 Mạng Hồi Qui Nhảy (Recurrent Skipping Networks - RSNs) : 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** [0], 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 (3.2.2) 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 [25]) 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 kế 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 [26] 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 kế cận từ bộ ba ban đầu. Một cải tiến khác của GAT bằng cách thêm thông tin của vector nhúng quan hệ là KBAT [22]. 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 [0]) .

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 tối đa hóa xác xuất 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 mất mát dựa trên lề .

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

  Ở phương pháp Cực đại hóa xác xuất tái cấu trúc cạn (maximize edge reconstruct probability), một đỉnh nhúng tốt sẽ cực đại hóa xác xuất 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 sư 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 này, các đỉnh được nhúng vào một không gian ngữ nghĩa ẩn nơi khoảng cách giữa cách 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 [1] tổng hợp và trình bày lại ở bảng 3.1. 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. Các phương pháp sử dụng kỹ thuật bước nhảy ngẫu nhiễn 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.

Nhóm phương pháp	Danh mục con	Ưu điểm	Nhược điểm
Phân rã ma trận	Đồ thị toán tử Laplace Eigenmap	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	Cực đại hóa xác xuất 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	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	Nhúng đồ thị dựa trên không gian ẩn	Phép nhúng có thể giải thích được	Khó điều chỉnh lựa chọn phân bố Yêu cầu một lượng lớn dữ liệu huấn luyện
	Nhúng kết hợp ngữ nghĩa	Tận dụng nhiều thông tin nguồn một cách tự nhiên	
Học sâu	Sử dụng bước ngẫu nhiên	Hiệu quả và nhanh chóng	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	Không phải trích đặc trưng	Chi phí tính toán cao

Bảng 3.1: Bảng so sánh ưu và nhược điểm của các kỹ thuật nhúng 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.

Tuy nhiện, nhược điểm của nhóm phương pháp học sâu đối với nhúng đồ thị là nếu coi các đỉnh và cạnh trong đồ thị như một lưới để thực hiện tích chập thì sẽ không đúng với bản chất của đồ thị vì không đảm bảo cấu trúc không gian của đồ thị. Mô hình mạng chú ý trên đồ thị dựa trên cơ chế chú ý giúp tổng hợp thông tin dựa vào các trọng số chú ý của một thực thể này đối với một thực thể khác. Chúng tôi cho rằng đây là hướng nghiên cứu tương tự như cơ chế ghi nhớ và chú ý của con người [0] : Ghi nhớ là quá trình liên kết những gì chưa biết với những gì đã biết [0]. Vì vậy đây là hướng nghiên cứu chúng tôi chọn trong các nhóm phương pháp trên.

3.2.3 Cơ chế cộng tác đa đỉnh chú ý

Cơ chế chú ý đa đỉnh (multi-head attention) là một phương pháp con của mô hình Transformer [25] được nhóm tác giả đề xuất năm 2017 giúp thể hiện sự quan trọng của một từ với các từ khác trong một câu. Cơ chế chú ý đa đỉnh có thể đại diện cho bất kỳ phép tính tích chập nào [0]. Trong phần này chúng tôi sẽ trình bày chi tiết về cơ chế chú ý đa đỉnh cũng như cải tiến mới nhất trên cơ chế chú ý đa đỉnh [16] được chúng tôi gọi là *cộng tác đa đỉnh chú ý* (collaborate multi-head attention).

Cơ Chế Chú Ý (Attention Mechanism)

Ta gọi $\mathbf{X} = \left\{\overrightarrow{x_1}, \overrightarrow{x_2}, ..., \overrightarrow{x_{N_x}}\right\}$ và $\mathbf{Y} = \left\{\overrightarrow{y_1}, \overrightarrow{y_2}, ..., \overrightarrow{y_{N_y}}\right\}$ là các ma trận nhúng đầu vào, với mỗi dòng i^{th} hay j^{th} trong ma trận $\mathbf{X}$ hay $\mathbf{Y}$ là một vector nhúng $\overrightarrow{x_i} \in \mathbb{R}^{1\times D_{\text{in}}}$, $\overrightarrow{y_j} \in \mathbb{R}^{1\times D_{\text{in}}}$. 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 như công thức sau :

$$\mathbf{H} = \text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\Big(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\Big)\mathbf{V} \tag{3.1}$$

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 vector thể hiện quá trình tham số hóa để biến đổi vector nhúng đầu vào D_{in} chiều thành vector nhúng D_k hoặc $D_{\text{attention}}$ chiều. $\mathbf{Q}\mathbf{K}^T$ 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 thể hiện độ lớn xác xuất của từng giá trị chú ý đối với 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) .

Cơ Chế Chú Ý Đa Đỉnh (Multi-Head Attention Mechanism)

Tương tự như cơ chế chú ý ở trên, cơ chế chú ý đa đỉnh (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 đỉnh khác nhau giúp ổn định trong quá trình huấn luyện. Cơ chế chú ý đa đỉnh 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 :

$$\begin{aligned} \mathbf{X}' &= \left(\Big\Vert_{h=1}^{N_{\text{head}}} \mathbf{H}^{(h)} \right) \mathbf{W}^O \\ &= \left(\Big\Vert_{h=1}^{N_{\text{head}}} \text{Attention}(\mathbf{X}\mathbf{W}_Q^{(h)}, \mathbf{Y}\mathbf{W}_K^{(h)}, \mathbf{Y}\mathbf{W}_V^{(h)}) \right) \mathbf{W}^O \end{aligned} \tag{3.2}$$

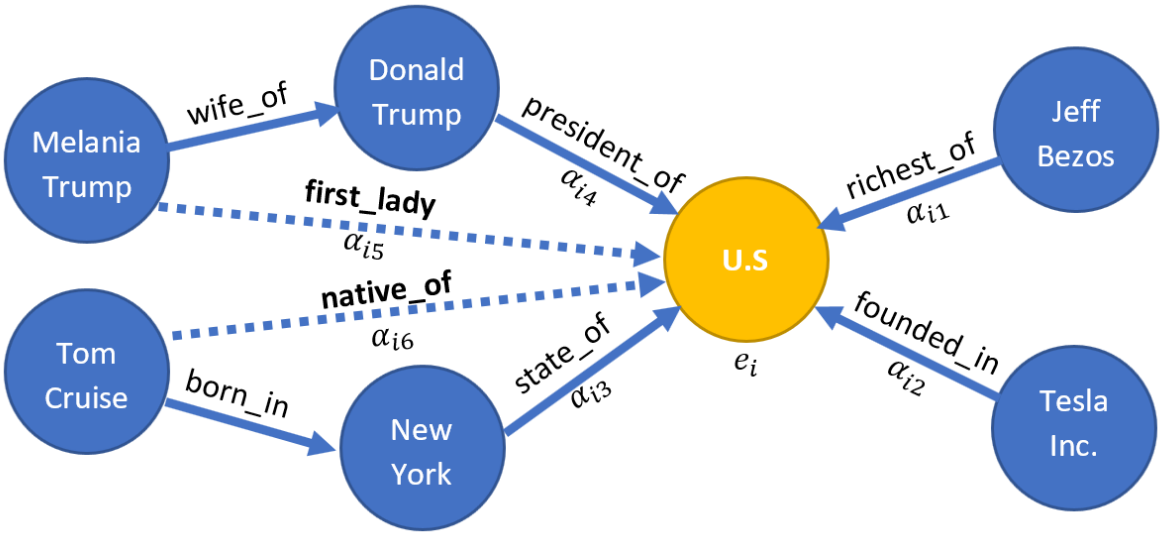
Trong đó các ma trận trọng số $\mathbf{W}_Q^{(h)}$, $\mathbf{W}_K^{(h)} \in \mathbb{R}^{D_{\text{in}} \times D_k}$ và $\mathbf{W}_V^{(h)} \in \mathbb{R}^{D_{\text{in}} \times D_{\text{attention}}}$ thuộc vào từng lớp chú ý $h \in [N_{\text{head}}]$ khác nhau. $\mathbf{W}^O \in \mathbb{R}^{N_{\text{head}}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

Lớp cộng tác đa đỉnh chú ý

[0]

3.2.4 Mạng đồ thị chú ý

Với kết quả cải thiện đáng kể của *cơ chế chú ý* trong xử lý ngôn ngữ tự nhiên, nó còn được nghiên cứu để áp dụng vào xử lý ảnh [0]. Cơ chế chú ý có thể đại diện cho bất kỳ phép tính tính chập nào [0], vì vậy cơ chế chú ý đã được nghiên cứu để áp dụng vào các mô hình nhúng đồ thị tri



Hình 3.9: Đồ thị tri thức và các hệ số chú ý chuẩn hóa của thực thể

thức thay cho một phương pháp tính chập như Mạng Đồ Thị Tích Chập (GCNs [20]). Ở phần này chúng tôi sẽ trình bày chi tiết về cách cơ chế chú ý ở 3.2.3 được áp dụng vào việc nhúng đồ thị theo phương pháp Mạng Đồ Thị Chú Ý (Graph Attention Network - GAT [26]).

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) trong không gian : $\mathbf{E} = \left\{\overrightarrow{e_1}, \overrightarrow{e_2}, ..., \overrightarrow{e_{N_e}}\right\}$. Mục tiêu của mô hình là biến đổi thành ma trận nhúng đầu ra mới $\mathbf{E}'' = \left\{\overrightarrow{e''_1}, \overrightarrow{e''_2}, ..., \overrightarrow{e''_{N_e}}\right\}$ với khả năng tổng hợp thông tin nhúng từ các thực thể lân cận; $\mathbf{E} \in \mathbb{R}^{N_e \times D_{\text{in}}}$ 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 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 3.2.3, 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. 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 $\mathbf{X} = \mathbf{Y} = \mathbf{E}$ (3.2.3) 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 3.2 vào đồ thị được mô tả như sau :

$$e_{ij} = f_{\text{mask attention}}(\mathbf{W}\overrightarrow{e_i}, \mathbf{W}\overrightarrow{e_j}) \tag{3.3}$$

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 $\overrightarrow{e_i}$ 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 :

$$\overrightarrow{e_i} \xrightarrow{f^{(1)}_{\text{mask attention}}} \overrightarrow{e_i'} \xrightarrow{f^{(2)}_{\text{mask attention}}} \overrightarrow{e_i''} \tag{3.4}$$

Ở quá trình chú ý đa đỉnh đầu tiên ($f^{(1)}_{\text{mask attention}}$), mô hình sẽ tổng hợp thông tin từ các thực thể lân cận và ghép chồng lên nhau để tạo ra vector $\overrightarrow{e_i'}$, với $\overrightarrow{e_i'} \in \mathbb{R}^{1\times D'}$. Ở bước thứ hai ($f^{(2)}_{\text{mask attention}}$), lớp chú ý đa đỉnh đã đỉnh 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 $\overrightarrow{e_i'}$ tiếp tục được xem là vector nhúng đầu vào để biến đổi thành vector nhúng $\overrightarrow{e_i''}$ cuối cùng với $\overrightarrow{e_i''} \in \mathbb{R}^{1\times D''}$.

Đầu tiên, giống như cơ chế chú ý 3.1, mỗi vector nhúng sẽ được nhân với một ma trận trọng số $\mathbf{W} \in \mathbb{R}^{D_k \times D_{\text{in}}}$ 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 :

$$\overrightarrow{h_i} = \mathbf{W}\overrightarrow{e_i} \tag{3.5}$$

khi đó $\overrightarrow{e_i} \in \mathbb{R}^{D_{\text{in}} \times 1} \rightarrow \overrightarrow{h_i} \in \mathbb{R}^{D_k \times 1}$

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 :

$$e_{ij} = \Big(\text{LeakyReLU}\Big(\overrightarrow{\mathbf{a}}^T[\overrightarrow{h_i}||\overrightarrow{h_j}]\Big)\Big) \tag{3.6}$$

với $.^T$ là phép chuyển vị, $||$ là phép ghép. Tương tự 3.1, tuy nhiên thay vì thực hiện tính tích vô hướng thì ta sử dụng một *cơ chế chú ý chung* (shared attentional mechanism) $\overrightarrow{\mathbf{a}} : \mathbb{R}^{D_k} \times \mathbb{R}^{D_k} \rightarrow \mathbb{R}$ để tính hệ số chú ý. Như đã trình bày ở 3.3, 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 :

$$\alpha_{ij} = \frac{\exp\Big(\text{LeakyReLU}\Big(\overrightarrow{\mathbf{a}}^T[\overrightarrow{h_i}||\overrightarrow{h_j}]\Big)\Big))}{\sum_{k \in \mathcal{N}_i} \exp\Big(\text{LeakyReLU}\Big(\overrightarrow{\mathbf{a}}^T[\overrightarrow{h_i}||\overrightarrow{h_k}]\Big)\Big))} \tag{3.7}$$

Ở bước này, mô hình GAT tương tự như GCN [20], 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 :

$$\overrightarrow{e_i'} = \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}\overrightarrow{h_j}\right) \tag{3.8}$$

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 ($f^{(1)}_{\text{mask attention}}$ 3.4) đầu tiên của mô hình:

$$\overrightarrow{e_i'} = \Big\Vert_{h=1}^{N_{\text{head}}} \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}^h \mathbf{W}^h \overrightarrow{h_j}\right) \tag{3.9}$$

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 3.1 $\mathbf{W}^h$ 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 $\overrightarrow{e_i'} \in \mathbb{R}^{1\times D'}$ với $D' = N_{\text{head}} D_{\text{k}}$ tiếp tục được xem là vector đầu vào để thực hiện cơ chế chú ý. Tuy nhiên ở bước thứ hai ($f^{(2)}_{\text{mask attention}}$ 3.4) giá trị chú ý đa đỉnh sẽ được tính trung bình thay vì ghép chồng lên nhau theo công công thức sau :

$$\overrightarrow{e_i''} = \sigma\left(\frac{1}{N_{\text{head}}}\sum_{h=1}^{N_{\text{head}}}\sum_{j\in\mathcal{N}_i} \alpha_{ij}^h \mathbf{W}^h \overrightarrow{e_j'}\right) \tag{3.10}$$

3.2.5 Mô hình KBAT

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 3.9, Donald Trump vừa đóng là vai trò là tổng thống, vừa đóng vai trò là người chồng. Do đó, mô hình Nhúng đồ thị dựa trên chú ý - KBAT (graph attention based embeddings [22]) kết hợp thêm thông tin của quan *quan hệ và đặc trưng các đỉnh hàng xóm* vào trong cơ chú ý.

Mô hình sẽ biến đổi từ ma trận nhúng thực thể $\mathbf{E} = \left\{\overrightarrow{e_1}, \overrightarrow{e_2}, ..., \overrightarrow{e_{N_e}}\right\} \rightarrow \mathbf{E}'' = \left\{\overrightarrow{e''_1}, \overrightarrow{e''_2}, ..., \overrightarrow{e''_{N_e}}\right\}$, với $\mathbf{E} \in \mathbb{R}^{N_e \times D_{\text{in}}}$ và $\mathbf{E}'' \in \mathbb{R}^{N_e \times D''}$. Đồng thời biến đổi ma trận nhúng quan hệ $\mathbf{R} = \left\{\overrightarrow{r_1}, \overrightarrow{r_2}, ..., \overrightarrow{r_{N_r}}\right\} \rightarrow \mathbf{R}'' = \left\{\overrightarrow{r''_1}, \overrightarrow{r''_2}, ..., \overrightarrow{r''_{N_r}}\right\}$ với $\mathbf{R} \in \mathbb{R}^{N_r \times P_{\text{in}}}$ và $\mathbf{R}'' \in \mathbb{R}^{N_r \times P''}$. Tương tự như mô hình GAT đã trình bày ở mục 3.2.4, 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 KBAT sẽ ghép các vector nhúng thực thể và vector nhúng quan hệ theo cấu trúc như sau :

$$\overrightarrow{t_{ijk}} = \mathbf{W_1}[\overrightarrow{e_i}||\overrightarrow{e_j}||\overrightarrow{r_k}] \tag{3.11}$$

Với $\overrightarrow{t_{ijk}}$ là vector nhúng đại diện cho bộ ba $t^k_{ij} = (e_i, r_k, e_j)$ 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 [15].

Tương tự với công thức 3.7 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 :

$$\begin{aligned}\alpha_{ijk} &= \text{softmax}_{jk}(\text{LeakyReLU}(\mathbf{W_2}\overrightarrow{t_{ijk}})) \\ &= \frac{\exp\Big(\text{LeakyReLU}\Big(\mathbf{W_2}\overrightarrow{t_{ijk}}\Big)\Big)}{\sum_{n\in\mathcal{N}_i}\sum_{r\in\mathcal{R}_{in}}\exp\Big(\text{LeakyReLU}\Big(\mathbf{W_2}\overrightarrow{t_{inr}}\Big)\Big)}\end{aligned} \tag{3.12}$$

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 nối đỉnh gốc e_i với thực thể $e_n \in \mathcal{N}_i$. Tương tự công thức 3.8, các vector nhúng $\overrightarrow{t^k_{ij}}$ sẽ được thu nhỏ hoặc mở rộng khi

nhân với hệ số chú ý đã được chuẩn hóa :

$$\overrightarrow{e_i'} = \sigma\left(\sum_{j\in\mathcal{N}_i}\sum_{k\in\mathcal{R}_{ij}} \alpha_{ijk}\overrightarrow{t_{ijk}}\right) \tag{3.13}$$

Tương tự như công thức 3.9 của *cơ chế mặt nạ chú ý*, ta sẽ ghép N_{head} đỉnh chú ý lại với nhau để ổn định quá trình học :

$$\overrightarrow{e'}_i = \Big\Vert_{h=1}^{N_{\text{head}}} \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ijk}^{(h)}\overrightarrow{t_{ijk}^{(h)}}\right) \tag{3.14}$$

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 :

$$\mathbf{R}' = \mathbf{R}\mathbf{W}^R; \qquad \text{với: } \mathbf{W}^R \in \mathbb{R}^{P\times P'} \tag{3.15}$$

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 3.10 . 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 :

$$\overrightarrow{e_i''} = \sigma\left(\frac{1}{N_{\text{head}}}\sum_{h=1}^{N_{\text{head}}}\sum_{j\in\mathcal{N}_i}\sum_{k\in\mathcal{R}_{ij}} \alpha_{ijk}^{\prime(h)}\overrightarrow{t_{ijk}^{\prime(h)}}\right) \tag{3.16}$$

Với $\alpha_{ijk}^{\prime(h)}$ và $t_{ijk}^{\prime(h)}$ 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 KBAT đã áp dụng giống như mô hình GAT 3.2.4 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. Để giải quyết vấn đề này, mô hình 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 vào để đảo bảo quá thông tin nhúng khởi tạo trong quá trình huấn luyện :

$$\mathbf{H} = \mathbf{W}^E \mathbf{E} + \mathbf{E}'' \tag{3.17}$$

3.2.6 Huấn luyện

Mô hình mượn ý tưởng từ hàm chấm điểm tịnh tiến (translational scoring function) từ mô hình TransE [15]

Our model borrows the idea of a translational scoring function from (Bordes et al., 2013), which learns embeddings such that for a given valid triple $t_{ij}^k = (e_i, r_k, e_j)$, the condition $\vec{h_i} + \vec{g_k} \approx \vec{h_j}$ holds, i.e., e_j is the nearest neighbor of e_i connected via relation r_k.

Specifically, we try to learn entity and relation embeddings to minimize the L1-norm dissimilarity measure given by $d_{t_{ij}} = \left|\left|\vec{h_i} + \vec{g_k} - \vec{h_j}\right|\right|_1$.

We train our model using hinge-loss which is given by the following expression

$$L(\Omega) = \sum_{t_{ij} \in S} \sum_{t'_{ij} \in S'} \max\{d_{t'_{ij}} - d_{t_{ij}} + \gamma, 0\} \tag{3.18}$$

where $\gamma > 0$ is a margin hyper-parameter, S is the set of valid triples, and S' denotes the set of invalid triples, given formally as

$$S' = \underbrace{\{t^k_{i'j} | e'_i \in \mathcal{E} \setminus e_i\}}_{\text{thay thế thực thể đầu}} \cup \underbrace{\{t^k_{ij'} | e'_j \in \mathcal{E} \setminus e_j\}}_{\text{thay thế thực thể đuôi}} \tag{3.19}$$

where ω^m represents the $m - th$ convolutional filter, ω is a hyper-parameter denoting number of filters used, $*$ is a convolution operator, and $\mathbf{W} \in \mathbb{R}^{\Omega k \times 1}$ represents a linear transformation matrix used to compute the final score of the triple. The model is trained using soft-margin loss as

$$\mathcal{L} = \sum_{t^k_{ij} \in \{S \cup S'\}} \log(1 + exp(l_{t^k_{ij}}.f(t^k_{ij}))) + \frac{\lambda}{2} \parallel \mathbf{W} \parallel_2^2 \tag{3.20}$$

where $l_{t^k_{ij}} = \begin{cases} 1 & \text{for } t^k_{ij} \in S \\ -1 & \text{for } t^k_{ij} \in S' \end{cases}$

Chương 4

Kết quả thực nghiệm và phân tích

sdfsdf

4.1 Tập dữ liệu

Tập dữ liệu bao gồm :

Dataset	Entities	Relations	Edges			
			Training	Validation	Test	Total
WN18RR	40,943	11	86,835	3034	3134	93,003
FB15k-237	14,541	237	272,115	17,535	20,466	310,116

Bảng 4.1: Mô tả tập dữ liệu

4.2 Phương pháp đánh giá

sdfsdf

	WN18RR					FB15K-237				
			Hits@N					Hits@N		
	MR	MRR	@1	@3	@10	MR	MRR	@1	@3	@10
TransE	7000	0.444	41.2	47	50.4	7000	0.444	41.2	47	50.4
ConvKB	7000	0.444	41.2	47	50.4	7000	0.444	41.2	47	50.4
Mô hình của chúng tôi										
AnyBURL	7000	<u>0.444</u>	41.2	47	<u>50.4</u>	7000	0.444	41.2	47	50.4
CGAT	7000	**0.444**	41.2	**47**	50.4	7000	0.444	41.2	47	50.4

Bảng 4.2: Kết quả thực nghiệm trên tập WN18RR và FB15K-237. Kết quả tốt nhất được **bôi đen** và kết quả tốt thứ hai được <u>gạch chân</u>

4.3 Phương pháp huấn luyện

sdfsdf

4.4 Kết quả và phân tích

sdfsdf

Chương 5

Kết luận

EXPERIMENTS RESULTS AND ANALYSIS
Dùng lện

Danh mục công trình của tác giả

1. Tạp chí ABC
2. Tạp chí XYZ

Tài liệu tham khảo

Đồ thị tri thức

[0] Bojchevski, Aleksandar et al. "Netgan: Generating graphs via random walks". In: *arXiv preprint arXiv:1803.00816* (2018).

[1] Cai, Hongyun, Zheng, Vincent W, and Chang, Kevin Chen-Chuan. "A comprehensive survey of graph embedding: Problems, techniques, and applications". In: *IEEE Transactions on Knowledge and Data Engineering* 30.9 (2018), pp. 1616–1637.

[2] Cao, Shaosheng, Lu, Wei, and Xu, Qiongkai. "Grarep: Learning graph representations with global structural information". In: *Proceedings of the 24th ACM international on conference on information and knowledge management*. 2015, pp. 891–900.

[0] Cordonnier, Jean-Baptiste, Loukas, Andreas, and Jaggi, Martin. "On the relationship between self-attention and convolutional layers". In: *arXiv preprint arXiv:1911.03584* (2019).

[0] Dettmers, Tim et al. "Convolutional 2d knowledge graph embeddings". In: *arXiv preprint arXiv:1707.01476* (2017).

[0] Google. *Introducing the Knowledge Graph: things, not strings*. 2020 (truy cập ngày 27/08/2020). URL: `https://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html`.

[3] Goyal, Palash and Ferrara, Emilio. "Graph embedding techniques, applications, and performance: A survey". In: *Knowledge-Based Systems* 151 (2018), pp. 78–94.

[0] Grover, Aditya and Leskovec, Jure. "node2vec: Scalable feature learning for networks". In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 2016, pp. 855–864.

[0] Guo, Lingbing, Sun, Zequn, and Hu, Wei. "Learning to exploit long-term relational dependencies in knowledge graphs". In: *arXiv preprint arXiv:1905.04914* (2019).

[0] Ivanov, Sergey and Burnaev, Evgeny. "Anonymous walk embeddings". In: *arXiv preprint arXiv:1805.11921* (2018).

[0] Ji, Shaoxiong et al. "A survey on knowledge graphs: Representation, acquisition and applications". In: *arXiv preprint arXiv:2002.00388* (2020).

[0] McKee, Douglas. *What are mind-blowing facts about human memory?* 2020 (truy cập ngày 26/08/2020). URL: `https://www.quora.com/What-are-mind-blowing-facts-about-human-memory`.

[4] Mousavi, Seyedeh Fatemeh et al. "Hierarchical graph embedding in vector space by graph pyramid". In: *Pattern Recognition* 61 (2017), pp. 245–254.

[0] Perozzi, Bryan, Al-Rfou, Rami, and Skiena, Steven. "Deepwalk: Online learning of social representations". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2014, pp. 701–710.

[5] Rossi, Andrea et al. "Knowledge Graph Embedding for Link Prediction: A Comparative Analysis". In: *arXiv preprint arXiv:2002.00819* (2020).

[0] Tang, Jian et al. “Line: Large-scale information network embedding”. In: *Proceedings of the 24th international conference on world wide web*. 2015, pp. 1067–1077.

[0] Ugander, Johan et al. “The anatomy of the facebook social graph”. In: *arXiv preprint arXiv:1111.4503* (2011).

[0] Việt-Nam, Bộ Y Tế. *Trí nhớ và chú ý*. 2020 (truy cập ngày 26/08/2020). URL: `https://healthvietnam.vn/thu-vien/tai-lieu-tieng-viet/bac-si-tam-ly/tri-nho-va-chu-y`.

[0] Vu, Thanh 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)*. 2019, pp. 2180–2189.

[0] Wang, Daixin, Cui, Peng, and Zhu, Wenwu. “Structural deep network embedding”. In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 2016, pp. 1225–1234.

Phương pháp AnyBURL

[6] Meilicke, Christian et al. “Anytime Bottom-Up Rule Learning for Knowledge Graph Completion.” In: *IJCAI*. 2019, pp. 3137–3143.

Phương pháp CGAT

[15] Bordes, Antoine et al. “Translating embeddings for modeling multi-relational data”. In: *Advances in neural information processing systems*. 2013, pp. 2787–2795.

[16] Cordonnier, Jean-Baptiste, Loukas, Andreas, and Jaggi, Martin. "Multi-Head Attention: Collaborate Instead of Concatenate". In: *arXiv preprint arXiv:2006.16362* (2020).

[17] He, Kaiming et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[18] Hopfield, John J. "Hopfield network". In: *Scholarpedia* 2.5 (2007), p. 1977.

[19] Jégou, Simon et al. "The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2017, pp. 11–19.

[20] Kipf, Thomas N and Welling, Max. "Semi-supervised classification with graph convolutional networks". In: *arXiv preprint arXiv:1609.02907* (2016).

[21] LeCun, Yann et al. "Object recognition with gradient-based learning". In: *Shape, contour and grouping in computer vision*. Springer, 1999, pp. 319–345.

[22] Nathani, Deepak et al. "Learning attention-based embeddings for relation prediction in knowledge graphs". In: *arXiv preprint arXiv:1906.01195* (2019).

[23] Nguyen, Dai Quoc et al. "A novel embedding model for knowledge base completion based on convolutional neural network". In: *arXiv preprint arXiv:1712.02121* (2017).

[0] Ramachandran, Prajit et al. "Stand-alone self-attention in vision models". In: *arXiv preprint arXiv:1906.05909* (2019).

[25] Vaswani, Ashish et al. "Attention is all you need". In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[26] Veličković, Petar et al. “Graph attention networks”. In: *arXiv preprint arXiv:1710.10903* (2017).

[0] Weng, Lilian. “Attention? Attention!” In: *lilianweng.github.io/lil-log* (2018). URL: `http://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html`.

[27] Yang, Zhilin et al. “Xlnet: Generalized autoregressive pretraining for language understanding”. In: *Advances in neural information processing systems*. 2019, pp. 5753–5763.

Phụ lục A

Kết quả training

Đây là phụ lục.

Phụ lục B

Tập dữ liệu

Đây là phụ lục 2.